МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ВІСНИК
ЧЕРКАСЬКОГО ДЕРЖАВНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ

Науковий збірник

Том 30,
№ 3. 2025

ЧЕРКАСИ
2025

ISSN: 2306-4412
E-ISSN: 2708-6070

Засновник:
Черкаський державний технологічний університет

Рік заснування: 1996
Рекомендовано до друку та поширення
через мережу Інтернет Вченою радою
Черкаського державного технологічного університету
(протокол № 2 від 15 вересня 2025 р.)

Державна реєстрація: Ідентифікатор медіа R30-04613.
Рішення Національної ради України з питань телебачення і радіомовлення
№ 1916, протокол № 17 (30.05.2024).

Науковий збірник входить до переліку наукових фахових видань України
Категорія «Б». Спеціальність:
технічні фізико-математичні (Наказ Міністерства освіти і науки України № 886 від 02.07.2020):
113 Прикладна математика, 121 Інженерія програмного забезпечення, 122 Комп'ютерні науки, 123 Комп'ютерна інженерія, 125 Кібербезпека та захист інформації,
126 Інформаційні системи та технології, 131 Прикладна механіка, 132 Матеріалознавство,
133 Галузеве машинобудування, 151 Автоматизація та комп'ютерно-інтегровані технології,
152 Метрологія та інформаційно-вимірювальна техніка,
172 Телекомунікації та радіотехніка; технічні (Наказ Міністерства освіти і науки України № 1188 від 24.09.2020): 101 Екологія, 161 Хімічні технології та інженерія

Науковий збірник представлено у міжнародних наукометричних базах даних, репозитаріях та пошукових системах: Bielefeld Academic Search Engine (BASE), Crossref, CiteFactor, Litmaps, Ulrich's Periodicals Directory, WorldCat,
J-Gate, Open Ukrainian Citation Index (OUCI),
Наукова періодика України, Національна бібліотека України імені В. І. Вернадського (НБУВ),
Dimensions, UCSB Library, Google Академія, German Union Catalogue of Serials (ZDB),
Бібліотека Університету Осло, Бібліотека Університету Халла,
SOLO - Search Oxford Libraries Online, Інститут Європейського університету, Бібліотека Лейпцизького університету (UBL), Бібліотека Кембриджського університету

Адреса редакції:
Черкаський державний технологічний університет
18006, бульв. Шевченка, 460, м. Черкаси, Україна
E-mail: info@bulletin-chstu.com.ua
https://bulletin-chstu.com.ua/uk

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
CHERKASY STATE TECHNOLOGICAL UNIVERSITY

BULLETIN
OF CHERKASY STATE TECHNOLOGICAL UNIVERSITY

Scientific Collection

Volume 30,
No. 3. 2025

CHERKASY
2025

ISSN: 2306-4412
E-ISSN: 2708-6070

Founder:
Cherkasy State Technological University

Year of foundation: 1996
Recommended for printing and distribution
via the Internet by the Academic Council
of Cherkasy State Technological University
(Minutes No. 2 of September 15, 2025)

State Registration: Media identifier R30-04613.
Decision of the National Council of Television and Radio Broadcasting of Ukraine
No. 1916, Minutes No. 17, dated 30.05.2024.

The scientific collection is included in the list of Professional Scientific Publications of Ukraine
Category "B". Specialties: technical, physical and mathematical (Order of the Ministry of Education and Science of Ukraine No. 886 of 02.07.2020): 0541 Mathematics,
0612 Database and network design and administration,
0613 Software and applications development and analysis, 0714 Electronics and automation,
0715 Mechanics and metal trades, 0716 Motor vehicles, ships and aircraft,
0788 Inter-disciplinary programmes and qualifications involving engineering, manufacturing and construction; technical (Order of the Ministry of Education and Science of Ukraine No. 1188 of 24.09.2020): 0521 Environmental sciences, 0522 Natural environments and wildlife,
0711 Chemical engineering and processes

The scientific collection is presented in the international scientometric databases, repositories and scientific systems: Bielefeld Academic Search Engine (BASE), Crossref, CiteFactor, Litmaps,
Ulrich's Periodicals Directory, WorldCat, J-Gate,
Open Ukrainian Citation Index (OUCI),
Scientific Periodicals of Ukraine, Vernadsky National Library of Ukraine (VNLU), Dimensions,
UCSB Library, Google Scholar, German Union Catalogue of Serials (ZDB), University of Oslo Library,
University of Hull Library, SOLO - Search Oxford Libraries Online, European University Institute,
Leipzig University Library (UBL), Cambridge University Library

Editors office address:
Cherkasy State Technological University
18006, 460 Shevchenko Blvd., Cherkasy, Ukraine
E-mail: info@bulletin-chstu.com.ua
https://bulletin-chstu.com.ua/en

Редакційна колегія

Editorial Board

Зміст / Contents

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.42:004.75
DOI: 10.62660/bcstu/3.2025.10

Framework for the comparative analysis of networking software libraries and its application to C++ networking solutions

Yehor Hrushevyy*
Master
Taras Shevchenko National University of Kyiv
01033, 60 Volodymyrska Str., Kyiv, Ukraine
https://orcid.org/0009-0005-7695-1990

Kostiantyn Zhereb
PhD in Physics and Mathematics, Assistant
Taras Shevchenko National University of Kyiv
01033, 60 Volodymyrska Str., Kyiv, Ukraine
https://orcid.org/0000-0003-0881-2284

Abstract. The objective evaluation of networking libraries remains a critical challenge in software engineering, as inconsistent methodologies and application-specific code often obscure meaningful performance and usability differences. The aim of this study was to conduct a comprehensive comparison of three C++ networking libraries (Boost.Asio, Boost.Beast, and Poco.Net) based on the proposed universal framework. The research methodology included the development of a universal comparative framework, experimental implementation of software prototypes, load testing to collect quantitative indicators, and a structured qualitative assessment based on a number of defined criteria. The framework encompassed both quantitative metrics (throughput, latency and resource consumption) and qualitative criteria (Application Programming Interface ergonomics, feature completeness, stability, cross platform compatibility and development complexity), with all libraries assessed under identical software conditions. Using each library, two C++ client-server prototypes were implemented: the RESTful service ResourceMonitor and the real time streaming application GameOfLifeStreaming. The project structure was unified to eliminate variability in application logic and to focus analysis exclusively on library behaviour. Boost.Asio demonstrated advantages in latency sensitive scenarios and scenarios requiring fine grained control. Boost.Beast offered effective HTTP support with minimal performance overhead but limited protocol coverage. Poco.Net achieved the lowest memory footprint and maintained stable performance at high loads while supporting the widest range of protocols, albeit at the expense of higher latency and lower raw throughput, and requiring more complex configuration. The practical contribution lies in the reusable framework for evaluating other networking libraries, the integration of created components into diverse development workflows, and the provided guidelines for choosing the most appropriate library

Keywords: application performance evaluation; load testing; client-server architecture; quantitative and qualitative metrics collection; testbed configuration; modular architecture; throughput and latency measurement

INTRODUCTION

In modern software engineering, objectively comparing and benchmarking libraries is essential to isolate the true impact of each implementation on performance, resource usage and developer ergonomics. Without a rigorous framework, application-specific logic or measurement inconsistencies can obscure the real strengths and weaknesses of candidate libraries. Numerous prior studies have addressed the challenge of selecting

Article's History: Received: 07.04.2025; Revised: 10.08.2025; Accepted: 15.09.2025.

Suggested Citation:
Hrushevyy, Ye., & Zhereb, K. (2025). Framework for the comparative analysis of networking software libraries and its application to C++ networking solutions. *Bulletin of Cherkasy State Technological University*, 30(3), 10-23. doi: 10.62660/bcstu/3.2025.10.

*Corresponding author

suitable software libraries by proposing structured methodologies and metricbased evaluation frameworks.

A variety of academic studies have also examined C++ networking stack. However, a comprehensive, side-by-side evaluation of Boost.Asio, Boost.Beast and Poco.Net within full-fledged client-server applications has yet to be conducted. E. Kadusic *et al*. (2022) investigated IPv6 address validation and security support in both Boost.Asio and Poco.Net, demonstrating that while both libraries offer flexible Application Programming Interfaces (APIs) for Internet of Things (IoT) device integration, their memory-handling strategies differ substantially, with Asio requiring more manual buffer management and Poco.Net leveraging hingher-level abstractions for certificate validation. Y. Pasichnyk (2022) and D. Butynets (2023) compared synchronous, asynchronous and hybrid server designs in pure C++ – often contrasting raw POSIX system calls against Asio's eventdriven model – and concluded that hybrid approaches can outperform purely threaded or purely asynchronous solutions under continuous stress, though each paradigm exhibits unique trade-off in central processing unit (CPU) utilisation and latency.

Several studies have employed Boost.Asio – and, to a lesser extent, Boost.Beast and Poco.Net – within broader application contexts, yet without focusing on detailed comparative analysis between the libraries. S. Kayum *et al*. (2020) integrated Boost.Asio into a fault-tolerant implementation of a massively parallel seismic processing framework, demonstrating its reliability and scalability in high performance computing environments with tens of thousands of cores. J. Ardarve *et al*. (2019) leveraged Poco.Net for a privacyenforcement agent under COPPA constraints, noting the library's ease of message-passing but not quantifying its runtime characteristics. In addition, cross-language comparisons shed light on C++ networking's relative strengths and weaknesses. C. Pflugfelder (2022) evaluated asynchronous programming across IoT platforms in C, C++, Rust, Lua and Python on ESP32 devices, finding that C++ (using Asio) consistently achieved lower end-to-end latency at the cost of increased code complexity.

Taken together, these works highlight the maturity and versatility of high-level C++ networking libraries but also reveal significant gaps. Prior research has tended to focus on individual libraries, specific protocols, or broader language-level comparisons, without offering a unified and empirical evaluation of Boost.Asio, Boost.Beast and Poco.Net across consistent, real-world client-server scenarios. Although isolated benchmarks have shed light on the strengths of these libraries, no prior work has systematically compared their behaviour under identical conditions using reproducible, modular setups.

In parallel to the focused evaluation of C++ networking libraries, the broader ecosystem of software comparison and communication frameworks offers valuable insights that inform this methodology. P. Sai *et al*. (2024) developed a Python-based real-time task manager with Psutil and Tkinter for system monitoring, illustrating the importance of modular, responsive data collection. Y. Baddi *et al*. (2023) proposed an SDNbased multicast protocol for largescale IoT deployments, demonstrating how flexible group communication can optimise latency and scalability. A. Samoydiuk & O. Ostapchuk (2024) investigated asynchronous programming techniques in smart-home management systems, demonstrating significant performance gains in high-load scenarios. Collectively, these works reinforce the value of a dedicated, structured framework for reproducible, multi-dimensional evaluation of networking solutions across diverse environments.

The aim of this study was to evaluate and contrast three widely used C++ networking libraries – Boost.Asio, Boost.Beast, and Poco.Net using a consistent comparative framework. To achieve this aim, the following tasks were undertaken. First, to define a consistent set of criteria and metrics suitable for the comparative analysis of networking libraries, and to apply them to the assessment of the selected libraries via two representative client-server applications. Second, to collect both quantitative and qualitative data through controlled experiments and structured observation. Third, to analyse the resulting data to produce a structured comparison across key dimensions, including performance, API ergonomics, feature completeness, reliability, portability, and community support.

MATERIALS AND METHODS

The comparative framework was centred on seven core dimensions (performance and efficiency, architecture and API, functionality, developer ergonomics, reliability and stability, portability and compatibility, community support), each of which was chosen to capture a distinct facet of a networking library's suitability for real-world deployment. The performance and efficiency dimension was used to quantify raw throughput and latency under representative workloads, to ensure that libraries could satisfy application-level service-level agreements. Metrics such as requests per second, mean response time, and high-percentile latencies were selected because they directly reflect user-perceived responsiveness and system scalability. The architecture and API design were assessed for clarity, consistency and modularity of the library's interfaces; well-designed APIs reduce cognitive load and lower the risk of integration errors, so measures of API surface area and adherence to established design idioms formed this dimension. Functionality examined protocol coverage, extensibility mechanisms and built-in utilities – libraries that natively support a broader set of protocols can address a wider range of use-cases without resorting to custom implementations.

Developer ergonomics were evaluated, recognising that ease of learning and use heavily influences time-to-market and long-term maintenance costs; to capture this, the complexity of API required for common tasks,

the prevalence of synchronous versus asynchronous patterns, and the availability of diagnostic messages were estimated. Reliability and stability were used to measure a library's fault-handling model, resource-cleanup guarantees and behaviour under stress, since production services must recover gracefully from network anomalies. Portability and compatibility were used to gauge the effort needed to build and deploy across different platforms and toolchains, ensuring that a library can +serve diverse environments from embedded devices to cloud services. Finally, community support was assessed to reflect each library's ecosystem health-release cadence, issue-tracker responsiveness and third-party contributions – as sustained maintenance and peerreviewed improvements are vital for longlived systems.

To operationalise the performance and efficiency dimension, a set of quantifiable indicators (such as latency, throughput, memory footprint and CPU utilisation) was defined and uniform procedures were established for collecting the relevant measurements using industry-standard tools, including ApacheBench and Windows Performance Monitor. A central tenet of the framework was the isolation of the networking layer via a uniform client-server interface. Each library under test was encapsulated within a plugin module conforming to the same IClient/IServer API. Buildtime selection of the desired networking module was handled by build variables, to guarantee that all other application logic remained constant. This modular architecture not only ensured that observed differences in behaviour could be attributed solely to the libraries themselves but also provided a reusable pattern for applying the framework to other programming languages or network-oriented libraries with minimal additional effort.

All experiments and prototype development were conducted on a Windows 11 environment using Visual Studio Code as the primary IDE. The build system was configured using CMake version 3.25.1-msvc1, targeting the Microsoft C++ compiler cl.exe version 19.35.32216.1 with the C++20 standard. Python 3.13.0 was used for the data-collection and database agents. Source control and version management were performed using Git version 2.39.0.windows.2. The networking libraries under investigation included Boost version 1.86.0 (specifically the Boost.Asio and Boost.Beast components) and POCO C++ Libraries version 1.14.1 (Poco::Net and associated modules). Additional dependencies included the nlohmann/json library version 3.11.3 for JSON serialisation and deserialisation, and raylib version 5.5 for graphical rendering in the streaming prototype. Performance metrics were collected using Windows Performance Monitor version 10.0. HTTP benchmarking was performed using ApacheBench from the apache2-utils package version 2.4.52-1ubuntu4.14 on the Ubuntu 22.04.5 LTS guest system with access via VirtualBox 7.0.14. Source code size was evaluated using cloc (Count Lines of Code) version 1.90. SQLite version 3.45.3 was employed as the database backend.

Two complete client-server applications were developed to exercise and compare the network libraries under study. These applications were chosen to represent distinct categories of network communication – request-response and real-time streaming – which enabled the evaluation of each library across contrasting usage scenarios. The first, ResourceMonitor, was designed as a distributed monitoring system for computer resource usage. It comprised five interchangeable modules (Fig. 1): a data-collection agent, a database-interaction agent, a central server, a command-line client, and a web-based client. The data-collection agent periodically sampled CPU, memory, disk and network metrics and transmitted them via HTTP to the central server, which in turn validated and forwarded the metrics to the database agent. Both user-facing clients (console and Vue.jsbased GUI) issued HTTP requests to the server and rendered the returned JSON data. To isolate networking behaviour, all three C++ libraries (Boost.Asio, Boost.Beast and Poco.Net) were encapsulated behind a common IClient/IServer interface and were selected at build time via CMake variables.

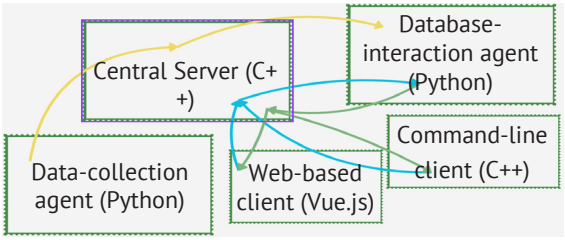


Figure 1. Components of ResourceMonitor
Source: created by the authors

For the Boost.Asio variant, the code was divided into three logical parts – client-specific, server-specific, and shared between client and server. A shared core defined HttpRequest and HttpResponse classes inheriting from a base HttpMessage, responsible for constructing and parsing the textual representation of HTTP messages. The client class managed a single io_context, driving asynchronous domain name system (DNS) resolution, connection establishment, request transmission and response reception – each operation performed by explicit calls to Asio's asynchronous functions and coordinated on a dedicated Input/Output (I/O) thread. Request cancellation was supported through Asio timers. Correspondingly, the server class used an Asio acceptor to receive incoming transmission control protocol (TCP) connections, then issued asynchronous read and write operations to parse requests, invoke the database agent, and send back JSONformatted responses.

In contrast, the Boost.Beast implementation leveraged Beast's higher-level HTTP constructs. A single Session class encapsulated both client and server logic, using Beast's http::async_read/async_write to process entire messages in one operation. The Beast client simply composed a http::request, submitted it via a Beast stream,

and awaited a http::response object, while the server acceptor spawned a new Session for each connection, automatically handling header framing, body buffering and protocol compliance. By reusing the same Boost-Common primitives, such as the I/O service singleton and the database request manager, both the Asio and Beast implementations shared identical lowerlevel infrastructure.

The Poco.Net version took a yet more high-level approach. In ResourceMonitor, HTTP functionality was provided by HTTPClientSession, HTTPRequest and HTTPResponse classes. The client constructed and dispatched requests, while the server was realised via Poco's HTTPServer framework, with a custom HTTPRequestHandlerFactory producing HTTPRequestHandler instances per request. Custom threadpool manager was introduced to overcome Poco's default pool limitations, queueing tasks and dispatching them only when worker threads became available. Data collection was handled by a Python-based agent that ran as a background service on each monitored host. This agent periodically invoked native system APIs to sample CPU, memory, disk and network metrics, packaged the results as JSON, and transmitted them via HTTP requests to the central server. The choice of Python for this component was motivated by its rich ecosystem for system monitoring and its rapid development cycle, which permitted the agent to be implemented and refined with minimal effort.

Central server performed validation of user requests and database responses and organised communication between other components. It had a list of configurable options such as port, number of threads or coordinates of database agent. Its implementation included controller with high-level logic and configuration parser. The central server's networking layer was exclusively targeted and measured during the performance tests. Python database agent exposed a simple Flaskbased HTTP interface for accepting and querying metric data. This agent mapped each request to operations on an SQLite database. The database schema – initialised on first run – comprised a primary mapping of IP addresses to internal host IDs and a series of timeseries tables (e.g. CPU, memory, disk and network usage), each keyed by host ID and timestamp. By isolating persistence logic in Python, the central C++ server and client components could rely on a uniform HTTP API without embedding any database-specific code. User interaction in ResourceMonitor was facilitated by two distinct frontends. A web dashboard, implemented in Vue.js, issued asynchronous HTTP GET requests to REST endpoints, parsed the returned JSON payloads, and rendered real-time charts to visualise resource trends. In parallel, a C++ command-line client offered a text-based interface: users configured server address, logging level and output paths through Boost.Program_options; entered commands such as request or cancel; and received formatted responses via IClient/IServer interface employed by the underlying networking module.

The second prototype, GameOfLifeStreaming, simulated John Conway's Game of Life and transmitted its evolving universe state as a real-time data stream to multiple clients. In this simpler architecture, a C++ server carried the simulation logic and periodically emitted the entire universe state over UDP multicast. Clients subscribed to the multicast group, reassembled each complete simulation state – hereafter referred to as a "frame" – and rendered it using the lightweight raylib graphics library. Networking for the GameOfLifeStreaming prototype was encapsulated behind similar IClient/IServer abstraction layer employed in the ResourceMonitor application. Each library's implementation – Boost.Asio, Boost.Beast and Poco.Net – was compiled as a separate static networking module, selectable at build time via a single CMake flag. The core networking interface provided methods to initialise connections, send and receive raw "frames", and register callback handlers for events such as "frame" arrival or connection errors.

Under Boost.Asio and Poco.Net – both of which provide native support for UDP features – the networking layer allowed the application to open a UDP multicast socket bound to the configured group address and port, and to spawn an I/O thread to asynchronously receive datagrams. Incoming UDP packets were passed into a "frame" reassembly component, which detected complete universe "frames" by scanning for the terminating newline marker. In contrast, Boost.Beast does not natively support UDP or multicast communication. To preserve the integrity of the comparison – and ensure that each library is evaluated solely on its own capabilities – it was opted not to introduce third-party dependencies. Instead, Beast's performance was assessed using a semantically equivalent TCP-based streaming approach via WebSocket. The IClient implementation used Beast's websocket::async_read and websocket::async_write calls to receive and send text "frames", while the server maintained a pool of session objects, each driving its own asynchronous read/write operations.

The server comprised three cooperating C++ classes: a coordinator that initialised and gracefully shut down the application, a configuration parser built with Boost.Program_options that validated command-line arguments (e.g. universe dimensions, tick rate, thread count, logging options), and a simulation class that applied Conway's rules to update a two-dimensional grid. Each update generated a concise text "frame" – prefixed by three-digit width and height specifiers and followed by a stream of # and space characters – then delivered it to the networking layer. Each project's source tree was partitioned into a root directory containing global configuration files (e.g. gitignore, CMakeLists.txt) and subdirectories for the client, server, utility and networking components. The CMake build scripts were structured hierarchically: a toplevel CMakeLists.txt defined the overall project, required CMake version and C++ standard, and delegated control to per-module CMakeLists.txt

files. In each module, source and header files were added to either a static library or an executable target.

A central config.cmake file provided paths to external dependencies (Boost, Poco, raylib, nlohmann/json) and compile-time options to select the desired networking implementation. By toggling a single CMake variable, the build system included exactly one of the three static networking libraries – Asio, Beast or Poco.Net – thereby producing three variants of each prototype without altering application code. Common C++ utility components were factored into their own static library to avoid duplication. It included logging system, built on the Strategy pattern, input/output helper functions for argument parsing and working with JSON, and a Singleton pattern.

The performance and stability of each prototype variant were evaluated under controlled laboratory conditions to ensure reproducibility. A uniform hardware and network testbed was established comprising three hosts (Fig. 2): two Windows 11 machines and Ubuntu 22.04 virtual guest (hosted via VirtualBox). Ubuntu guest was configured with a bridged network adapter, thereby obtaining a distinct IP address on the same LAN as its host. Entries for test nodes were added to each machine's hosts file to permit name-based addressing. Windows Firewall rules were adjusted on the physical hosts to allow incoming TCP, UDP and ICMP traffic from the other test nodes.

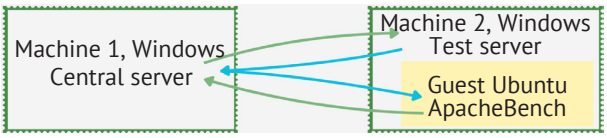


Figure 2. Testbed for ResourceMonitor
Source: created by the authors

HTTP performance was measured using ApacheBench (invoked via the ab command-line tool), selected for its wide adoption and ability to generate configurable request loads. For each test scenario, ab was invoked with parameters -n (total requests) and -c (concurrent clients) against the ResourceMonitor HTTP endpoint, yielding per-request latency and throughput statistics. Streaming latency in GameOfLifeStreaming was measured using embedded timestamps: the server prefixed each multicast payload with its local time, and the client computed time by comparing receipt time against this timestamp. System resource usage on Windows hosts was sampled with Performance Monitor (perfmon), capturing Processor Time, Working Set and Private Working Set at one-second intervals. These metrics were exported as time series for analysis of CPU and memory trends under sustained load.

Two test harnesses were devised to exercise each networking library. For the HTTP workload, the server endpoint under test accepted GET requests, redirected them to test server (that replaced database component and returned preconfigured JSON bodies) and returned its answers to client. ApacheBench was used to generate load from a separate client host, varying concurrency parameters (c = 5, 100, 500) and total request counts (n = 1,000, 15,000, 10,000). Server builds were configured with either two or six threads. Key metrics captured for comparison were requests per second (throughput), mean time per request (ms), 90th percentile latency (ms), 99th percentile latency (ms) and system resource utilisation (CPU and memory footprint) monitored via Windows Performance Monitor.

The streaming scenario measured end-to-end latency and resource consumption when broadcasting "Game of Life" "frames" to many subscribers. Tests spawned 5, 100, 1,000 and 15,000 logical clients within a single process, each collecting 1,000 timestamped "frames" to compute mean latency. Server configurations again alternated between two and six threads (for Beast implementation, not applicable for Asio and Poco.Net implementations), and Performance Monitor logged CPU and memory trends throughout each run.

In addition to empirical performance testing, each networking library was evaluated using a structured comparison across a set of other criteria. These included functionality breadth, developer usability, codebase complexity, runtime reliability, portability and compatibility, and community support. The purpose of this complementary analysis was to capture aspects not visible through benchmarking alone but essential for real-world applicability. A small set of objective questions (Table 1) was defined for each dimension. Binary questions were scored as yes/no (1 or 0), while others were evaluated on a 5-point scale (0, 0.25, 0.5, 0.75, 1). The raw sums of these perquestion scores yielded a dimension-specific total, which can be compared directly across libraries.

Table 1. Questions for qualitative evaluation

No.	Dimension	Question	Scoring
1.1	Architecture and API	Does the library separate connection establishment, data transfer and message processing into distinct modules?	yes/no (1/0)
1.2	Architecture and API	Are all public classes and methods documented with parameter descriptions, behavioural notes and exception details?	yes/no (1/0)
1.3	Architecture and API	How well does the API adhere to standard C++ naming and style conventions?	scale (0-1)
1.4	Architecture and API	How easily can a new protocol be integrated into the API without modifying the library's source?	scale (0-1)
2.1	Functionality	Rate amount of protocols the library support out of the box.	scale (0-1)

Continued Table 1.

No.	Dimension	Question	Scoring
2.2	Functionality	Rate the adequacy of built-in utilities for SSL/TLS certificate management, DNS resolution and connection recovery.	scale (0-1)
2.3	Functionality	How straightforward is it to switch between synchronous and asynchronous modes within the same API?	scale (0-1)
2.4	Functionality	Does the library guarantee threadsafe use of core objects across multiple threads?	yes/no (1/0)
3.1	Developer Ergonomics	Estimate the number of lines of code required to perform basic tasks.	scale (0-1)
3.2	Developer Ergonomics	Rate the quality of error diagnostics (exception detail, error codes).	scale (0-1)
3.3	Developer Ergonomics	How comprehensive are the "end-to-end" usage examples in the documentation?	scale (0-1)
3.4	Developer Ergonomics	Rate learning curve.	scale (0-1)
4.1	Reliability and Stability	Are open sockets and timers automatically closed on unexpected errors?	yes/no (1/0)
4.2	Reliability and Stability	Does the library support automatic reconnection following a lost connection?	yes/no (1/0)
4.3	Reliability and Stability	Rate the library's resilience under sustained load (no hangs, no memory leaks).	scale (0-1)
4.4	Reliability and Stability	Does the library provide built-in timeout mechanisms and deadlock prevention?	yes/no (1/0)
5.1	Portability and Compatibility	Rate the number of supported platforms (Windows, Linux, macOS, embedded).	scale (0-1)
5.2	Portability and Compatibility	Rate the number of external dependencies required for core functionality.	scale (0-1)
5.3	Portability and Compatibility	How simple is integration with common build systems?	scale (0-1)
5.4	Portability and Compatibility	Can both 32bit and 64bit builds be produced without source modifications?	yes/no (1/0)
5.5	Portability and Compatibility	Are prebuilt binaries provided for multiple target platforms?	(yes/no)
6.1	Community Support	How many active contributors are listed in the project repository?	scale (0-1)
6.2	Community Support	What is the average response time to issues in the official tracker?	scale (0-1)
6.3	Community Support	How frequently are new releases published (monthly, quarterly, biannual, annual)?	scale (0-1)
6.4	Community Support	Are there official discussion forums or mailing lists for support?	yes/no (1/0)
6.5	Community Support	Rate the number of commits in the project repository.	scale (0-1)

Source: created by the authors

RESULTS AND DISCUSSION

The following results reflect the application of the comparative analysis framework to three C++ networking libraries – Boost.Asio, Boost.Beast, and Poco.Net – across two representative real-world scenarios. Each metric presented here corresponds to one of the framework's seven evaluation dimensions. In line with the performance and reliability dimensions, all three implementations demonstrated robust results, with no HTTP requests failing during any of the tests. Under a two-thread configuration and lower load (Table 2), Boost.Asio achieved the highest throughput, followed by Boost.Beast. As the load increased, the relative ranking remained unchanged, with Asio leading, Beast in the middle, and Poco.Net trailing (Table 3).

Table 2. Throughput and latency for 15,000 requests and concurrency level 100, ResourceMonitor

Metric	Boost.Asio		Boost.Beast		Poco.Net	
	2 threads	6 threads	2 threads	6 threads	2 threads	6 threads
Requests per second	114.75	66.03	34.08	58.27	88.06	162.11
Mean time per request (ms)	43.57	75.73	146.71	85.82	56.78	30.84
90^{th} percentile latency (ms)	51	79	227	104	91	41
99^{th} percentile latency (ms)	268	1,099	1,135	141	255	59

Source: created by the authors

Table 3. Throughput and latency for 1,000 requests and concurrency level 5, ResourceMonitor

Metric	Boost.Asio		Boost.Beast		Poco.Net	
	2 threads	6 threads	2 threads	6 threads	2 threads	6 threads
Requests per second	351.04	485.57	354.32	333.77	103.48	276.49
Mean time per request (ms)	284.87	205.94	282.23	299.61	966.34	361.67
90th percentile latency (ms)	412	263	354	370	1 245	430
99th percentile latency (ms)	993	404	669	1,046	3,658	562

Source: created by the authors

Under high-concurrency conditions (Table 4), the disparities between the libraries became more pronounced. Boost.Asio preserved its performance advantage, while Poco.Net exhibited a sharp decline in throughput.

Table 4. Throughput and latency for 10,000 requests and concurrency level 500, ResourceMonitor

Metric	Boost.Asio		Boost.Beast		Poco.Net	
	2 threads	6 threads	2 threads	6 threads	2 threads	6 threads
Requests per second	447.15	478.66	386.72	357.15	106.83	266.66
Mean time per request (ms)	1,118.21	1,044.57	1 292.92	1,399.96	4,680.42	1,875.07
90th percentile latency (ms)	1,312	1,355	1,657	1,974	6,110	1,998
99th percentile latency (ms)	2,037	2,121	2,878	4,211	7,877	2,504

Source: created by the authors

Increasing the thread count to six did not alter this ranking, although Poco.Net outperformed the others at low concurrency levels (Table 2). Latency was characterised by mean response time as well as the 90th and 99th percentiles. For two threads (Table 2), Asio exhibited the lowest average latency, marginally ahead of Beast; Poco.Net's latency increased noticeably with higher concurrency (Tables 3, 4). At six threads, Poco.Net delivered the best mean latency under light loads but performed less favourably at higher loads, while Asio consistently outperformed Beast. The 90th percentile latencies followed similar patterns, with the gap between Beast and Poco.Net narrowing; at the 99th percentile, Poco.Net occasionally outperformed Beast, though Asio remained fastest. In the GameOfLifeStreaming tests (Table 5), UDP multicast support in Asio and Poco.Net did not yield a marked reduction in latency, indicating that client-side processing of many concurrent streams dominated delay rather than transport speed. Beast's WebSocketbased, percli-ent session model sometimes achieved lower average latency. Even at high client counts, absolute latencies remained modest, and increasing server thread count further reduced delays.

Table 5. Latency values, GameOfLifeStreaming

Clients number	Boost.Asio	Boost.Beast, 2 threads	Boost.Beast, 6 threads	Poco.Net
5	1.24	5.81	5.34	1.31
100	4.41	9.33	8.85	4.61
1,000	9.96	16.81	13.17	10.76
15,000	40.93	47.39	35.68	39.74

Source: created by the authors

CPU utilisation profiles across all ResourceMonitor test runs (Table 6) remained within expected bounds, exhibiting neither gradual increases nor pronounced spikes, thereby indicating stable processor load for each implementation. Working Set and Private Working Set metrics showed no evidence of memory leaks: values fluctuated steadily around a constant mean with no upward trend. In shorter runs Asio and Beast occasionally displayed a logarithmic growth pattern, but extended-duration tests revealed prompt resource reclamation. At peak load Asio and Beast consumed up to 52.5 MB Working Set compared to Poco.Net's 9.8 MB, a difference negligible on modern servers but potentially significant in resource-constrained environments.

Table 6. Resources usage metrics, ResourceMonitor

Metric	Boost.Asio		Boost.Beast		Poco.Net	
	2 threads	6 threads	2 threads	6 threads	2 threads	6 threads
Processor time, %, max	17	19	17	20	18	20
Processor time, %, mean	6	8	7	8	8	10
Working set, max, MB	48.3	52.5	47.3	49.4	8.1	9.8
Working set, mean, MB	22.1	27.4	20.8	24.5	7.2	7.6
Working set – private, max, MB	44.8	49.3	43.9	45.2	2.2	4.2
Working set – private, mean, MB	19.2	21.6	18.7	42.11	1.7	3.7

Source: created by the authors

In the GameOfLifeStreaming scenario (Table 7), Asio and Poco.Net benefited from UDP multicast to maintain low server resource consumption. By contrast, Beast's perclient sessions imposed greater CPU and memory overhead; with 15,000 concurrent clients, peak memory usage reached around 490 MB. The results of the qualitative evaluation are presented in Table 8. According to these results, Poco.Net leads in most dimensions.

Table 7. Resources usage metrics, GameOfLifeStreaming

Metric	Boost.Asio	Poco.Net	Boost.Beast	
			2 threads	6 threads
Processor time, %, max	6	15	73	98
Processor time, %, mean	4	7	41	45
Working set, max, MB	12.1	5.7	489.1	170.5
Working set, mean, MB	11.8	5.7	263.2	160.6
Working set – private, max, MB	5.8	0.8	260.9	153.2
Working set – private, mean, MB	5.7	0.7	241.7	142.1

Source: created by the authors

Table 8. Questions for qualitative evaluation

No.	Dimension	Question / Sum	Boost.Asio	Boost.Beast	Poco.Net
1.1	Architecture and API	Does the library separate connection establishment, data transfer and message processing into distinct modules?	1	1	1
1.2	Architecture and API	Are all public classes and methods documented with parameter descriptions, behavioural notes and exception details?	1	1	1
1.3	Architecture and API	How well does the API adhere to standard C++ naming and style conventions?	0.75	0.75	0.5
1.4	Architecture and API	How easily can a new protocol be integrated into the API without modifying the library's source?	0.25	0.25	0.75
1	**Architecture and API**	**Sum**	**3**	**3**	**3.25**
2.1	Functionality	Rate amount of protocols the library support out of the box.	0.25	0.25	1
2.2	Functionality	Rate the adequacy of built-in utilities for SSL/TLS certificate management, DNS resolution and connection recovery.	0.5	0.5	0.75
2.3	Functionality	How straightforward is it to switch between synchronous and asynchronous modes within the same API?	0.5	0.25	0
2.4	Functionality	Does the library guarantee threadsafe use of core objects across multiple threads?	1	1	1
2	**Functionality**	**Sum**	**2.25**	**2**	**2.75**
3.1	Developer Ergonomics	Estimate the number of lines of code required to perform basic tasks.	0.25	0.75	0.5
3.2	Developer Ergonomics	Rate the quality of error diagnostics (exception detail, error codes).	0.25	0.5	0.75

Continued Table 8.

No.	Dimension	Question / Sum	Boost.Asio	Boost.Beast	Poco.Net
3.3	Developer Ergonomics	How comprehensive are the "end-to-end" usage examples in the documentation?	0.25	0.5	0.75
3.4	Developer Ergonomics	Rate learning curve.	0.25	0.5	0.75
3	**Developer Ergonomics**	**Sum**	**1**	**2.25**	**2.75**
4.1	Reliability and Stability	Are open sockets and timers automatically closed on unexpected errors?	1	1	1
4.2	Reliability and Stability	Does the library support automatic reconnection following a lost connection?	0	0	1
4.3	Reliability and Stability	Rate the library's resilience under sustained load (no hangs, no memory leaks).	1	1	1
4.4	Reliability and Stability	Does the library provide built-in timeout mechanisms and deadlock prevention?	1	1	1
4	**Reliability and Stability**	**Sum**	**3**	**3**	**4**
5.1	Portability and Compatibility	Rate the number of supported platforms (Windows, Linux, macOS, embedded)?	1	1	1
5.2	Portability and Compatibility	Rate the number of external dependencies required for core functionality.	1	1	0.5
5.3	Portability and Compatibility	How simple is integration with common build systems?	0.75	0.75	0.5
5.4	Portability and Compatibility	Can both 32bit and 64bit builds be produced without source modifications?	1	1	1
5.5	Portability and Compatibility	Are prebuilt binaries provided for multiple target platforms?	1	1	1
5	**Portability and Compatibility**	**Sum**	**4.75**	**4.75**	**4**
6.1	Community Support	How many active contributors are listed in the project repository?	0.25	0.75	0.5
6.2	Community Support	What is the average response time to issues in the official tracker?	0.25	0.25	0.5
6.3	Community Support	How frequently are new releases published (monthly, quarterly, biannual, annual)?	0.5	0.5	0.75
6.4	Community Support	Are there official discussion forums or mailing lists for support?	1	1	1
6.5	Community Support	Rate the number of commits in the project repository.	1	1	0.5
6	**Community Support**	**Sum**	**3**	**3.5**	**3.25**
		Total Sum	**17**	**18.5**	**20**

Source: created by the authors

Boost.Asio offers the lowest level of abstraction and maximum flexibility by providing direct access to sockets, timers and file descriptors. Boost.Beast builds on top of Asio to raise the abstraction for specific application-level protocols (HTTP/HTTPS and WebSocket), but otherwise introduces no additional capabilities in connection management or I/O handling. By contrast, Poco.Net conceals system calls behind a well-structured, high-level API (factories, managers), which accelerates typical development tasks. The consistency of each library's API naturally correlates with its level of abstraction: Asio and Beast emphasise asynchronous operation, whereas Poco.Net favours synchronous patterns, which – as demonstrated by the measurements (Table 3, 4) – can be a notable drawback in latency-sensitive applications.

Boost.Asio supports raw TCP, UDP and ICMP sockets at the lowest level, with built-in asynchronous DNS resolution and timers. It accommodates both IPv4 and IPv6 and integrates readily with OpenSSL for SSL/TLS. However, it provides no application-level protocol implementations, requiring developers to implement or integrate additional libraries for HTTP, WebSocket, etc. Boost.Beast addresses this gap for HTTP and WebSocket, while preserving full access to Asio's low-level primitives; beyond these two protocols, no further application-level support is offered. Poco.Net provides the most comprehensive out-of-the-box functionality: TCP/UDP, IPv4/IPv6, DNS, HTTP/HTTPS, WebSocket, FTP, POP3 and SMTP, with SSL/TLS via NetSSL_OpenSSL.

Boost.Asio demands the steepest learning curve and development effort, owing to its low-level style and multitude of asynchronous operation calls. Documentation within Boost is well-structured but primarily APIcentric; example code is often focused on individual operations rather than complete solutions. Boost.Beast offers demonstration programs for HTTP and WebSocket

clients and servers, but error messages in both libraries tend to be technical, occasionally hindering rapid diagnosis. Poco.Net is designed for rapid prototyping, with extensive ready-made examples for common scenarios and more descriptive exception messages. Its principal downside is the larger surface area of external module dependencies. Code size, as measured by cloc (Table 9), reflects these trade-offs: Poco.Net exhibits big number of files and lines despite its high level of abstraction, owing to the necessity of additional handler and factory classes; Beast's WebSocket usage in GameOfLifeStreaming similarly increases its code footprint.

Table 9. Code size

Project	Boost.Asio		Boost.Beast		Poco.Net	
	Files	Lines	Files	Lines	Files	Lines
ResourceMonitor	15	1,165	13	782	17	929
GameOfLifeStreaming	5	320	9	788	7	561

Notes: this refers specifically to the size of the networking-related application code written using each library, rather than the size of the libraries themselves
Source: created by the authors

Boost.Asio employs both standard C++ exceptions (in synchronous code) and error_code objects (in asynchronous code) for error handling, with RAII ensuring proper resource cleanup. It is battle-tested in critical systems, although its asynchronous model can present complex debugging scenarios. Boost.Beast augments this with protocol-specific checks (e.g. header constraints) that improve resilience to malformed input or attacks, at the cost of additional exception handling. In Poco.Net, all errors derive from the hierarchical Poco::Exception base class, with built-in automatic reconnection mechanisms on connection loss. High-level interfaces in Poco.Net further reduce the incidence of low-level socket errors.

Boost.Asio and Boost.Beast are highly portable, being primarily header-only and depending only on Boost.System (and optionally OpenSSL). They compile cleanly with modern GCC, Clang and MSVC on Linux, Windows, macOS and many embedded environments. Poco, by contrast, is a modular library with various external components (OpenSSL, ODBC, etc.), which can complicate its build and configuration. Nonetheless, each Poco module remains independent, so unused dependencies need not be included. All three libraries support recent C++ standards and integrate smoothly with common build systems. In terms of startup speed and minimal external dependencies, Asio and Beast lead; Poco.Net offers a broader ecosystem at the cost of more involved setup.

All three libraries benefit from active ecosystems. Boost.Asio's repository exceeds 2,300 commits by around twenty core contributors. Boost.Beast has over 2,400 commits from 143 participants. Support is primarily via the Boost mailing lists and GitHub issues, with new Boost releases typically every four months. The Poco.Net include commits from almost 100 authors and more than 1,300 commits. Releases occur five to six times per year, announced on the official blog. Community engagement is available via GitHub, mailing lists and the blog. These repository statistics (Table 10) informed scoring of the Community Support dimension in the comparison framework (Table 8).

Table 10. GitHub repository metrics

Metric	Boost.Asio	Boost.Beast	Poco.Net
Commits	2,314	2,469	1,306
Contributors	19	143	97
Open issues	36	92	120
Releases frequency	4 months	4 months	2-2.5 months

Source: created by the authors

These results were subsequently considered in the context of prior research on software library evaluation and benchmarking. H. Anzt *et al.* (2019) proposed an automated benchmarking infrastructure for high-performance computing software. Their framework supports continuous performance testing and reproducibility by integrating benchmark execution, environment capture and visualisation via a web interface. The tool, originally developed for the Ginkgo library, demonstrates how such methodologies can contribute to sustainable scientific software by enabling community-driven performance monitoring.

Several works have also explored multiperspective and multicriteria evaluation models. K. Dawood *et al.* (2023) addressed the challenge of selecting open-source software in environments with heterogeneous user roles. They proposed a usability evaluation framework that integrates the Best Worst Method (BWM) and Group VIKOR techniques for weighting and ranking alternatives, validating their approach through a real-world case study. E. Denisova *et al.* (2024) introduced a reproducible usability testing framework tailored to medical software, with a focus on ensuring comparability across similar platforms. Their methodology, which

includes standardised tasks and objective performance measurements, reinforces the importance of formalised, domain-specific evaluation strategies. Although these studies cover a diverse range of software ecosystems, from web development to medical imaging and simulation, they collectively underscore the benefits of using formal methodologies and quantifiable criteria in the comparison of software libraries. Several works (Petty *et al.*, 2015; De La Mora & S. Nadi, 2018; Loja & Maita, 2024) also emphasise the importance of usability and performance metrics, the involvement of multiple stakeholder perspectives, and the potential for automation and reproducibility in evaluation processes.

The present work builds on these principles by introducing a structured framework for the empirical comparison of networking libraries, which is then applied to C++ solutions. M. Che & M. Tuo (2016) analysed asynchronous HTTP server designs with Winsock to avoid blocking and enhance responsiveness, while D. Mitrović *et al.* (2015) presented a scalable webagent middleware for automatic load balancing and fault tolerance. B. Amirkhanov *et al.* (2025) compared MQTT over TCP, MQTT over WebSocket, and HTTP in digital twin applications, emphasising the trade-offs between latency and stability. Finally, R. El-Hajj & S. Nadi (2020) introduced LibComp, an IntelliJ plugin enabling metricbased, inIDE comparison of Java libraries. F. Lu *et al.* (2019) observed that MongoDB's reliance on Boost.Asio, while beneficial for portability, introduced performance limitations under data-intensive workloads – limitations that were addressed by replacing Asio-based networking with a high-performance RDMA-capable transport layer. These studies underscore the versatility of Boost.Asio across domains but also highlight the need for a more systematic performance-oriented evaluation.

Several aspects of the findings resonate with and extend insights from both domain-specific benchmarking efforts and broader methodological research. Prior studies have emphasised the value of structured, metric-based evaluation frameworks in guiding software library selection across diverse contexts. For example, F. De La Mora & S. Nadi (2018) demonstrated how repository analytics can support informed library choice, highlighting the significance of popularity, security and performance indicators. M. Petty *et al.* (2015) proposed a formal assessment methodology for simulation frameworks, arguing that well-defined criteria are essential for objectivity and replicability. Multi-criteria evaluation models have also been introduced to address stakeholder diversity, such as the BWM and Group VIKOR integration by K. Dawood *et al.* (2023).

The results obtained in this study are in line with current trends in the use of smart technologies and modelling in the field of energy and network management. In particular, T. Nechaieva *et al.* (2025) emphasised the importance of comprehensive modelling of energy balances of local communities, taking into account environmental factors and the need to reduce greenhouse gas emissions. Their approach to building sustainable and decentralised energy systems complements the practical focus of this work's framework on flexibility and reuse. In turn, O. Turchyn (2025) demonstrated the effectiveness of using neural networks in optimising the management of technical systems, particularly in the oil and gas sector. The introduction of deep learning for adaptive control of sucker-rod pump installations confirms the relevance of using modern approaches to reduce energy consumption and increase efficiency, which also echoes this research findings on the resource stability of various libraries.

The empirical results obtained through the proposed framework align with earlier observations in the C++ networking domain while offering a more unified and reproducible basis for comparison. For instance, E. Kadusic *et al.* (2022) observed that Boost.Asio demands more manual buffer management compared to Poco.Net's higher-level abstractions. The hybrid server paradigms examined by Y. Pasichnyk (2022) and D. Butynets (2023) – which blended threaded and event-driven approaches – suggested performance gains under sustained stress; this was corroborated by the six-thread benchmarks. Finally, cross-language work by C. Pflugfelder (2022) highlighted C++/Asio's low end-to-end latency at the cost of code complexity – an observation mirrored in the latency profiles. Through this unified, empirical comparison, the study confirms earlier insights while providing the comprehensive, side-by-side performance evaluation of Boost.Asio, Boost.Beast and Poco.Net in production-style C++ applications.

The comparative results demonstrate that the proposed framework is capable of capturing both measurable performance characteristics and more qualitative aspects such as developer ergonomics and ecosystem maturity. By enforcing interface uniformity, and isolated testing environments, the framework enables meaningful side-by-side comparisons across different architectural paradigms and abstraction levels. This not only validates the methodology against existing benchmarking approaches in the literature, but also suggests its applicability beyond the C++ ecosystem to other networking stacks and programming environments.

Several directions exist for extending the present work. While the current evaluation was conducted within Windows 11 environment and a single Ubuntu 22.04.5 LTS guest, replicating the study across alternative operating systems, hardware configurations, and resource-constrained embedded environments could improve generalisability. Incorporating a broader set of HTTP benchmarking tools – such as wrk or JMeter – and varying request patterns (e.g., mixed GET/POST, chunked transfers) may help reveal additional behavioural traits under diverse conditions. Likewise, expanding the streaming scenario beyond Conway's Game of Life, by employing more demanding real-time protocols (e.g., QUIC, gRPC) or larger message formats, could provide further insights into performance and

resource trade-offs. Evaluating the impact of SSL/TLS handshake overheads, certificate validation delays, and network-level anomalies (e.g., packet loss or network jitter) could enrich the reliability analysis and inform security-sensitive use cases. Lastly, integrating formal metrics for code complexity, learning curve, and team adoption would complement the current qualitative assessment and support more rigorous comparison from a software engineering perspective.

CONCLUSIONS

This study conducted a structured comparison of three widely used C++ networking libraries – Boost.Asio, Boost.Beast, and Poco.Net – based on a unified evaluation approach. By encapsulating each candidate behind a uniform client-server interface and enforcing build-time selection, the methodology ensured that all non-networking logic remained invariant, enabling a trusted evaluation across performance, resource usage, and qualitative dimensions. The evaluation of network libraries from the perspective of not only performance indicators – such as throughput and latency – but also functionality, development complexity, reliability, and other relevant aspects, contributed to a more holistic and thorough analysis.

Applying this approach within two distinct C++ prototypes – a RESTful HTTP service and a real-time multicast streaming application – revealed clear trade-offs among Boost.Asio, Boost.Beast and Poco.Net. In the HTTP scenario, Boost.Asio delivered the highest request throughput and lowest average and tail latencies under modest threading configurations. Boost.Beast's higher-level framing engines delivered slightly lower performance (on average, up to 20% lower). Poco.Net, while generally trailing in raw throughput, exhibited the most stable hig-hpercentile latency characteristics and the smallest memory footprint, consuming less than a quarter of the working set required by its peers when serving equivalent loads. In the GameOfLifeStreaming prototype, neither Asio nor Poco.Net gained a decisive multicast-related advantage in end-to-end latency, indicating that the cost of managing large numbers of concurrent clients often outweighs the savings from UDP group transmission; Boost.Beast's WebSocket-based model occasionally matched or outperformed its UDP counterparts in the testbed.

Across both applications, CPU utilisation remained well within predictable bounds and showed no evidence of progressive degradation or leakage. Memory consumption in ResourceMonitor was markedly higher for the Asio and Beast implementations, reflecting their more granular buffer management and asynchronous contexts, whereas Poco.Net's synchronous architecture yielded consistently lower working and private working sets. In the GameOfLifeStreaming prototype, the Boost.Beast variant consumed significantly more resources due to the overhead of maintaining individual WebSocket sessions for each connected client.

The architecture and API of each library directly shaped these behaviours: Asio's minimal abstraction offered maximum flexibility at the cost of boilerplate and potential for misuse; Beast struck a middle ground by providing protocol-specific constructs atop Asio; and Poco.Net encapsulated broad protocol support behind a high-level, synchronous interface, trading off some performance for rapid development and ease of use. When evaluated using the qualitative scoring framework – covering modularity, documentation quality, extensibility, ergonomics and ecosystem maturity – Poco.Net achieved the highest total (20 points), followed by Beast (18.5 points) and Asio (17 points). Taken together, these findings suggest that no single library is universally "best". Instead, Asio is most appropriate where finegrained control and minimal runtime overhead are paramount, Beast is well-suited to HTTP or WebSocketcentric workloads that benefit from protocol-aware abstractions, and Poco.Net is favourable when development speed, multi-protocol support and constrained memory budgets prevail. The ability to swap libraries simply by toggling a buildtime flag, without any alteration to application code, proved invaluable for conducting this side-by-side analysis.

Looking forward, the comparative approach may be extended in several directions. Inclusion of additional protocols (for example, QUIC or gRPC), native SSL/TLS stacks, and alternative concurrency models will broaden its applicability. Evaluating across Linux, macOS and embedded environments will enhance generalisability, while more varied traffic patterns and security-focused tests will deepen understanding of each library's real-world robustness. Finally, integrating automated measures of developer productivity, code complexity and maintainability promises to complement the performance-driven insights with human-centred metrics, thereby closing the loop on fully rounded library evaluation.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Amirkhanov, B., Amirkhanova, G., Kunelbayev, M., Adilzhanova, S., & Tokhtassyn, M. (2025). Evaluating HTTP, MQTT over TCP and MQTT over WEBSOCKET for digital twin applications: A comparative analysis on latency, stability, and integration. *International Journal of Innovative Research and Scientific Studies*, 8(1), 679-694. doi: 10.53894/ijirss.v8i1.4414.

[2] Anzt, H., Chen, Y.-C., Cojean, T., Dongarra, J., Flegar, G., Nayak, P., Quintana-Ortí, E.S., Tsai, Y.M., & Wang, W. (2019). Towards continuous benchmarking: An automated performance evaluation framework for high performance software. In *PASC 2019: Proceedings of the platform for advanced scientific computing conference* (article number 9). New York: Association for Computing Machinery. doi: 10.1145/3324989.3325719.

[3] Ardarve, J., Finne, R., Hrustic, A., & Svennungsson, J. (2019). *Enforcing privacy requirements on oblivious network agents - a software solution of a type system generated from COPPA*. (Bachelor's thesis, Chalmers University Of Technology, Gothenburg, Sweden).

[4] Baddi, Y., Sebbar, A., Zkik, K., Maleh, Y., Bensalah, F., & Boulmalf, M. (2023). MSDN-IoT multicast group communication in IoT based on software defined networking. *Journal of Reliable Intelligent Environments*, 10, 93-104. doi: 10.1007/s40860-023-00203-x.

[5] Butynets, D. (2023). *Performance assessment of the different approaches to implementing network servers using C++ language*. (Bachelor's thesis, Ukrainian Ctholic University, Lviv, Ukraine).

[6] Che, M., & Tuo, M. (2016). Server program analysis based on HTTP protocol. *MATEC Web of Conferences*, 63, article number 05023. doi: 10.1051/matecconf/20166305023.

[7] Dawood, K.A., Zaidan, A.A., Sharif, K.Y., Ghani, A.A., Zulzalil, H., & Zaidan, B.B. (2023). Novel multi-perspective usability evaluation framework for selection of open source software based on BWM and group VIKOR techniques. *International Journal of Information Technology & Decision Making*, 22(1), 187-277. doi: 10.1142/s0219622021500139.

[8] De La Mora, F.L., & Nadi, S. (2018). An empirical study of metric-based comparisons of software libraries. In *Proceedings of the 14th international conference on predictive models and data analytics in software engineering* (pp. 22-31). New York: Association for Computing Machinery. doi: 10.1145/3273934.3273937.

[9] Denisova, E., Tiribilli, E., Luschi, A., Francia, P., Manetti, L., Bocchi, L., & Iadanza, E. (2024). Enabling reliable usability assessment and comparative analysis of medical software: A comprehensive framework for multimodal biomedical imaging platforms. *Health and Technology*, 14, 671-682. doi: 10.1007/s12553-024-00859-2.

[10] El-Hajj, R., & Nadi, S. (2020). LibComp: An IntelliJ plugin for comparing Java libraries. In *Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering* (pp. 1591-1595). New York: Association for Computing Machinery. doi: 10.1145/3368089.3417922.

[11] Kadusic, E., Zivic, N., Hadzajlic, N., & Ruland, C. (2022). The transitional phase of Boost. Asio and POCO C++ networking libraries towards IPv6 and IoT networking security. In *2022 IEEE international conference on smart internet of things (SmartIoT)* (pp. 80-85). Suzhou: IEEE. doi: 10.1109/smartiot55134.2022.00022.

[12] Kayum, S.N., Alsalim, H., Tonellot, T., & Momin, A. (2020). A fault tolerant implementation for a massively parallel seismic framework. In *2020 IEEE high performance extreme computing conference (HPEC)* (pp. 1-8). Waltham: IEEE. doi: 10.1109/hpec43674.2020.9286143.

[13] Loja, A., & Maita, T. (2024). Comparative evaluation of performance efficiency in terms of temporal behavior and resource utilization, according to the ISO/IEC 25,010 model, in a web application developed with Angular, React.js, and Vue.js. In G.F. Olmedo Cifuentes, D.G. Arcos Avilés & H.V. Lara Padilla (Eds.), *Emerging research in intelligent systems. CIT 2023. Lecture notes in networks and systems* (pp. 293-308). Cham: Springer. doi: 10.1007/978-3-031-52255-0_21.

[14] Lu, F., Fang, T., Zhang, Z., Li, S., Chen, J., An, H., & Han, W. (2019). Improving the performance of MongoDB with RDMA. In *IEEE international conference on high performance computing and communications (HPCC)* (pp. 1004-1010). Zhangjiajie: IEEE. doi: 10.1109/hpcc/smartcity/dss.2019.00144.

[15] Mitrović, D., Ivanović, M., Vidaković, M., & Budimac, Z. (2015). A scalable distributed architecture for Web-Based software agents. In M. Núñez, N. Nguyen, D. Camacho & B. Trawiński (Eds.), *Computational collective intelligence. Lecture notes in computer science* (pp. 67-76). Cham: Springer. doi: 10.1007/978-3-319-24069-5_7.

[16] Nechaieva, T., Teslenko, O., Trokhaniak, V., & Makarevych, S. (2025). Modeling energy balances of a community under increased energy independence and greenhouse gas reduction. *Machinery & Energetics*, 16(1), 104-116. doi: 10.31548/machinery/1.2025.104.

[17] Pasichnyk, Y. (2022). *Performance analysis of synchronous and asynchronous parallel network server implementations using the C++ language*. (Bachelor's thesis, Ukrainian Ctholic University, Lviv, Ukraine).

[18] Petty, M.D., Kim, J., Park, S., & Lee, S. (2015). A methodology for quantitative assessment of the features and capabilities of software frameworks for model composition. *International Journal of Modeling, Simulation, and Scientific Computing*, 7(1), article number 1541002. doi: 10.1142/s1793962315410020.

[19] Pflugfelder, C. (2022). *Asynchronous programs on IoT devices*. (Bachelor's thesis, Institute for Formal Methods of Computer Science, Stuttgart, Germany).

[20] Sai, P.C., Karthik, K., Prasad, K.B., Pranav, C.V.S., & Divya, K.V. (2024). Real-time task manager: A Python-based approach using Psutil and Tkinter. In *2024 8th international conference on computational system and information technology for sustainable solutions (CSITSS)* (pp. 1-6). Bengaluru: IEEE. doi: 10.1109/csitss64042.2024.10816758.

[21] Samoydiuk, A., & Ostapchuk, O. (2024). Optimizing high-load systems with asynchronous programming techniques. In *Modeling, control and information technologies: Proceedings of VII international scientific and practical conference* (pp. 130-131). Rivne: National University Of Water And Environmental Engineering. doi: 10.31713/mcit.2024.036.
[22] Turchyn, O. (2025). Introduction of neural network technologies to optimise control of suckerrod pump installation. *Machinery & Energetics*, 16(1), 32-42. doi: 10.31548/machinery/1.2025.32.

Фреймворк для порівняльного аналізу мережевих програмних бібліотек та його застосування до мережевих рішень C++

Єгор Грушевий

Магістр
Київський національний університет імені Тараса Шевченка
01033, вул. Володимирська, 60, м. Київ, Україна
https://orcid.org/0009-0005-7695-1990

Костянтин Жереб

Кандидат фізико-математичних наук, асистент
Київський національний університет імені Тараса Шевченка
01033, вул. Володимирська, 60, м. Київ, Україна
https://orcid.org/0000-0003-0881-2284

Анотація. Об'єктивне порівняння мережевих бібліотек залишається актуальним завданням у програмній інженерії, адже відсутність уніфікованих підходів та залежність від специфіки застосунків часто ускладнюють виявлення важливих відмінностей у продуктивності та зручності використання. Метою дослідження було здійснення комплексного порівняння трьох мережевих бібліотек C++ (Boost.Asio, Boost.Beast та Poco.Net) на основі запропонованого універсального фреймворку. Методологія дослідження охоплювала розробку універсального порівняльного фреймворку, експериментальну реалізацію програмних прототипів, проведення навантажувального тестування для збору кількісних показників та здійснення структурованої якісної оцінки за рядом визначених критеріїв. Створений фреймворк включає збір як кількісних (пропускна здатність, затримка, споживання ресурсів), так і якісних показників (зручність інтерфейсу прикладного програмування, функціональність, стабільність, кросплатформність, складність розробки). Особлива увага приділялась поведінці в умовах однакового програмного середовища. З використанням кожної з досліджуваних бібліотек, у межах роботи реалізовано два клієнт-серверні прототипи: RESTful-сервіс ResourceMonitor і застосунок з потоковим передаванням даних GameOfLifeStreaming. Варіанти реалізації уніфіковано в структурі проєкту, що забезпечило мінімізацію відмінностей на рівні логіки та дозволило сфокусуватись на властивостях самих бібліотек. Boost.Asio продемонстрував переваги у задачах із високими вимогами до затримок і необхідністю низькорівневого контрою. Boost.Beast виявився достатньо ефективним вибором для HTTP-застосунків, але з обмеженим переліком підтримуваних протоколів. Poco.Net найефективніше використовував ресурси, не сильно деградував при великих навантаженнях, дозволяє працювати з найбільшою кількістю протоколів, але показав гірші показники затримок і пропускної здатності, а також потребував складнішої конфігурації. Практична цінність роботи полягає у можливості повторного використання фреймворку для порівняння інших мережевих бібліотек, інтеграції створених компонентів в інші розробки, а також у сформованих висновках щодо доцільності застосування кожної з бібліотек

Ключові слова: оцінка продуктивності застосунків; навантажувальне тестування; клієнт-серверна архітектура; збір якісних і кількісних метрик; конфігурація тестового середовища; модульна архітектура; вимірювання пропускної здатності і затримок

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.056.55:004.451.7
DOI: 10.62660/bcstu/3.2025.24

Development and testing of the effectiveness of hybrid cryptographic algorithms for protecting personal data in mobile applications

Vitalii Yasenenko*
Master, Senior Software Developer
TP-Link
92618, 36 Technology Str., Irvine, USA
https://orcid.org/0009-0004-4801-9541

Abstract. With the rapid growth in the number of mobile applications and the volume of processed personal data, the need for effective means of the protection increased. The aim of the study was to develop hybrid cryptographic algorithms and to analyse the possibilities of the integration into mobile applications to enhance the protection of personal data. The study implemented hybrid cryptographic algorithms that combined symmetric and asymmetric encryption methods, modelled the operation using Python-based software modules, and assessed the efficiency considering the characteristics of the mobile environment. The main results showed that the combination of symmetric encryption algorithms Advanced Encryption Standard and ChaCha20 ensured complete data preservation during decryption and reduced the risk of compromise due to the two-layer encryption structure. It was also found that during testing of asymmetric methods, the Rivest-Shamir-Adleman algorithm successfully protected symmetric keys, and the Elliptic Curve Cryptography algorithm enabled both parties to compute a shared secret without transmitting the key itself, which improved resistance to interception. The results of the software implementation confirmed the identity of input and output data, demonstrating the reliability of the hybrid encryption approach. On Android platforms, Keystore, Bouncy Castle, Spongy Castle, and Conscrypt provided fast encryption using Advanced Encryption Standard (~275.6 milliseconds with hardware acceleration) and ChaCha20 (~2 milliseconds), as well as efficient key exchange via Elliptic Curve Cryptography (~15.8 milliseconds) compared to Rivest-Shamir-Adleman (~2,532.8 milliseconds), with support for post-quantum algorithms. On iPhone Operating System platforms, CryptoKit, CommonCrypto, and Open Secure Sockets Layer offered similar encryption speeds, while Secure Enclave optimised Elliptic Curve Cryptography, although post-quantum algorithms required additional optimisation. The hybrid approach reduced memory usage, making the algorithms effective for mobile devices. The obtained results could be used by mobile application developers to improve data protection in financial, medical, and corporate systems on smartphones running Android and iPhone Operating System platforms

Keywords: Advanced Encryption Standard; ChaCha20; Rivest-Shamir-Adleman; Elliptic Curve Cryptography; Android; iPhone Operating System

INTRODUCTION

In the context of the rapid spread of digital technologies, mobile devices became the primary environment for storing and transmitting confidential information, which necessitated the improvement of protection methods. Given the limited resources of mobile devices – particularly regarding energy consumption, performance, and memory – traditional cryptographic methods were not always effective in mobile environments. The problem was compounded by the fact that modern cyber threats use complex mechanisms to bypass protection, including traffic analysis, attacks on key infrastructure, data interception through compromised communication channels, or local interference. Since symmetric methods provided high speed but required

Article's History: Received: 19.03.2025; Revised: 29.07.2025; Accepted: 15.09.2025.

Suggested Citation:
Yasenenko, V. (2025). Development and testing of the effectiveness of hybrid cryptographic algorithms for protecting personal data in mobile applications. *Bulletin of Cherkasy State Technological University*, 30(3), 24-36. doi: 10.62660/bcstu/3.2025.24.

*Corresponding author

secure key exchange, and asymmetric ones ensured secure exchange but demanded more resources, there arose a need to develop hybrid solutions that would combine the advantages of both approaches.

For example, O. Borysenko & A. Tymoshenko (2024) focused on the risks of data leakage in cloud environments, emphasising the importance of encryption during storage and transmission, as well as the use of Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocols and access control to ensure the confidentiality of personal data. V. Semerenska (2025) analysed post-quantum cryptographic algorithms and concluded that lattice-based schemes – particularly Kyber and Nth-degree truncated polynomial ring unit – offered the best balance between performance and resistance to quantum attacks, also stressing the appropriateness of using hybrid solutions combining classical and post-quantum algorithms to protect critical systems. In addition, A. Sereda *et al*. (2023) analysed cryptographic algorithms used in mobile operating systems, reviewed approaches to protecting confidential information, disk space encryption, and end-to-end encryption, and proposed methods for evaluating the resilience and efficiency of cryptographic systems in mobile environments.

In turn, M.A. Sayed (2024) compared traditional cryptographic algorithms such as Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA), and hybrid models with machine learning-based models (autoencoders, Generative Adversarial Network, Random Forest) for encrypting personal data, demonstrating the advantages of the hybrid approach in balancing speed, security, and scalability. The study by G.A. Nwatuzie *et al*. (2025) proposed a hybrid cryptographic model combining AES, data encryption standard (DES), and Rivest's Cipher 6 to enhance encryption speed, scalability, and resistance to attacks in cloud environments, with support for automated key management and passwordless authentication. A. Muthaura & J. Kandiri (2024) developed and tested a cryptographic algorithm using Base64 and a fixed 512-bit length to enhance data protection, which showed optimal results in encryption speed, attack resilience, and plaintext vulnerability. Researchers H. Fan *et al*. (2022) also proposed a static "black-box" testing method for detecting the use of cryptographic algorithms in binary files without access to source code, revealing the widespread use of outdated and vulnerable algorithms such as Message Digest 4 and Secure Hash Algorithm 1.

The findings of R. Neve & R. Bansode (2024) presented the lightweight hybrid cryptographic algorithm Hybrid-SIMON-SPECKey, combining the SIMON round function and SPECK key schedule, which achieved 50% faster execution and 20% less memory usage compared to the original algorithms, while maintaining attack resilience under the avalanche effect criterion. The conclusions of A. Gour *et al*. (2024) confirmed the high effectiveness of the proposed hybrid cryptographic algorithm combining AES for symmetric encryption, Elliptic Curve Cryptography (ECC) for key exchange, and digital signature algorithm for digital signing in ensuring data confidentiality, integrity, and protection while maintaining computational efficiency. Furthermore, R.R. Pitale *et al*. (2024) underlined the importance of well-grounded cryptographic algorithm selection (including DES, 3DES, RSA, AES, International Data Encryption Algorithm, Blowfish, and homomorphic encryption) to ensure data confidentiality and integrity in the context of growing digital interaction, highlighting the role of algorithmic complexity and key management in countering modern threats.

The gaps in the cited studies lay in the insufficient focus on optimising hybrid cryptographic algorithms for mobile platforms, considering the hardware limitations, and in the limited analysis of the impact of such algorithms on performance and energy efficiency in mobile applications. In contrast, the present study focused on examining the prospects for applying hybrid cryptographic algorithms based on a mobile operating system to improve the protection of users' personal data. The research included the following objectives: to develop hybrid cryptographic algorithms combining symmetric and asymmetric encryption methods; to analyse the potential for implementing the developed algorithms in mobile operating system environments; and to carry out a comparative analysis of the effectiveness of hybrid solutions versus traditional cryptographic methods based on performance, energy consumption, and security levels.

MATERIALS AND METHODS

At the first stage of the study, a concept and basic implementation of hybrid cryptographic algorithms combining symmetric and asymmetric encryption methods were developed, taking into account the specifics of mobile environments. The main focus was placed on achieving a balance between cryptographic strength and computational efficiency. Within this stage, a justification for the hybrid approach was provided, where symmetric algorithms such as AES and ChaCha20 were used for primary data encryption, while asymmetric methods including RSA and ECC were used for secure key transmission. For clarity, a visualisation of the hybrid cryptographic algorithm using symmetric and asymmetric encryption was presented.

During testing, four separate software modules were created in Python using the cryptography library in the Visual Studio Code environment. The first programme implemented data encryption using AES-256 in Cipher Block Chaining mode with Public Key Cryptography Standard No. 7 padding. The second programme applied stream encryption using ChaCha20 with a unique nonce for sequential message protection. In turn, the third programme demonstrated encryption of the symmetric key using RSA with Optimal Asymmetric Encryption Padding and verified the correctness of its decryption. The fourth implemented key exchange via

the ECC-based Elliptic Curve Diffie-Hellman protocol, generating public and private key pairs for two parties and computing a shared secret.

At the second stage of the study, a comprehensive analysis of the integration of hybrid cryptographic algorithms into mobile applications was carried out, taking into account the peculiarities of the mobile environment based on comparative analysis methods and empirical evaluation of algorithm efficiency in a test environment. The focus was placed on assessing hardware constraints such as Central Processing Unit, Random Access Memory, energy consumption, and execution environments including Android Runtime (ART)/Dalvik and iOS runtime, as well as secure key storage capabilities (Keystore, Keychain). The analysis of mobile environment factors for cryptographic algorithms was conducted based on criteria such as computational resources, energy consumption, code execution characteristics across different mobile operating systems, support for libraries and Application Programming Interfaces, and secure key storage mechanisms (Ali *et al.*, 2025; Khalid *et al.*, 2025). For this purpose, a comparison of the performance of symmetric and asymmetric algorithms was conducted in terms of encryption time, key generation, memory usage, and energy consumption.

In addition, through structural-functional analysis, frameworks, and libraries supporting hybrid and post-quantum algorithm implementation were examined. For Android, Android Keystore and Bouncy Castle were considered, along with Spongy Castle and Conscrypt. Furthermore, the potential use of hybrid cryptography in combination with deep learning methods for detecting malware on Android was explored. In particular, a model combining a deep learning classifier with a cryptographic approach based on ECC and Blowfish was analysed. In the context of iOS, CryptoKit and CommonCrypto were reviewed. Additionally, tools such as Secure Enclave and OpenSSL were analysed. Based on this, a comparative analysis of mobile platforms was conducted using the above-mentioned tools, assessing the capabilities for implementing hybrid and post-quantum cryptographic algorithms. The analysis of iOS's comprehensive approach to data security covered information protection and SSL/TLS protocols (Kalphana *et al.* 2024; Prodduturi, 2025). Thus, this study carried out an analysis of the development, implementation, and adaptation of hybrid cryptographic algorithms, considering the characteristics of mobile platforms, which allowed for the identification of optimal combinations of encryption and key exchange methods to enhance the security and performance of mobile applications.

RESULTS

Development of hybrid cryptographic algorithms based on symmetric and asymmetric encryption methods

The protection of personal data in mobile applications requires a specific approach to cryptography, taking into account both the increased security requirements and the limited resources of devices. In this context, the development of hybrid cryptographic algorithms that combine symmetric and asymmetric encryption methods emerges as an effective solution that enables a balance between performance and cryptographic strength. The concept of hybrid encryption is based on the combination of two fundamental cryptographic principles. On the one hand, symmetric algorithms – such as AES (e.g., AES-256) or ChaCha20 – provide extremely fast encryption of large volumes of data with minimal processor load. On the other hand, asymmetric methods – particularly RSA or ECC – allow for the secure transmission of keys even in untrusted environments, as these methods rely on the computational complexity of factoring large numbers or computing the discrete logarithm on elliptic curves.

In the hybrid approach, data is encrypted using a symmetric method, which reduces processing time, while the symmetric key is encrypted using an asymmetric algorithm – and only the key is transmitted via an open channel. This combination minimises the risk of key interception without sacrificing speed, which is critically important in environments with limited mobile device performance. Such hybrid schemes can be effective in many security protocols, such as TLS; however, adapting these schemes to the specifics of mobile environments requires careful optimisation and testing. To better understand the main stages of hybrid encryption, a diagram is provided that illustrates the sequence of actions: from symmetric key generation to data encryption and the transmission of encrypted messages with a protected key, as well as subsequent decryption by the recipient (Fig. 1).

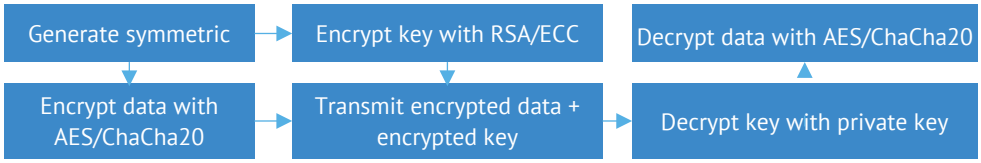


Figure 1. Scheme of operation of a hybrid cryptographic algorithm using symmetric and asymmetric encryption
Source: created by the author

The diagram illustrates the logic of combining symmetric and asymmetric encryption within a hybrid approach. One of the key stages in implementing a hybrid cryptographic algorithm is the correct organisation of the key generation and secure transmission process. In mobile environments, where resources

are limited, it is necessary to ensure not only cryptographic robustness but also the optimisation of this process in terms of performance and energy consumption. Within the hybrid approach, the symmetric key (to be used for the primary encryption of data) is generated on the sender's side using a cryptographically secure random number generator. This key is then encrypted with the recipient's public key (RSA or ECC) and transmitted as part of the message. That is, only the recipient, who possesses the corresponding private key, can decrypt the symmetric key and use it to decrypt the data. This approach effectively separates the tasks: symmetric algorithms perform the fast encryption/decryption of data, while asymmetric algorithms are used solely for the secure transmission of the key. Such a division of workload is critically important for mobile devices, where computational power and energy resources are limited.

Among symmetric cryptographic algorithms actively used in modern security systems, AES and ChaCha20 hold a special place. AES is a block-based symmetric encryption algorithm that provides reliable data protection and is widely used as an international standard. In contrast, ChaCha20 is a stream cipher optimised for high performance on devices without specialised cryptographic instructions, which makes it particularly effective in mobile applications with limited resources. Its strength lies in chaotic bit mixing and minimal use of complex operations. In Python, both algorithms can be implemented using the cryptography library, which supports flexible configuration of operating modes and encryption parameters (Fig. 2).

```
chacha_decryptor = Cipher(
    algorithms.ChaCha20(chacha_key, chacha_nonce),
    mode=None,
    backend=default_backend()
).decryptor()
aes_encrypted_restored = chacha_decryptor.update(chacha_encrypted)

print("🧩 [ChaCha20] Decrypted (AES encrypted message):", aes_encrypted_restored)

aes_decryptor = Cipher(
    algorithms.AES(aes_key),
    modes.CBC(aes_iv),
    backend=default_backend()
).decryptor()
padded_plaintext = aes_decryptor.update(aes_encrypted_restored) + aes_decryptor.finalize()

unpadder = padding.PKCS7(128).unpadder()
decrypted_message = unpadder.update(padded_plaintext) + unpadder.finalize()

print("✅ [AES-256] Final decrypted message:", decrypted_message)
```

Figure 2. Code snippet for implementing hybrid encryption using AES-256 and ChaCha20 in Python
Source: created by the author

The programme demonstrates the implementation of hybrid symmetric encryption, where data is first encrypted using AES-256 in Cipher Block Chaining mode with padding (extending the message to a multiple of the block length required for correct block cipher operation) according to Public Key Cryptography Standard No. 7, and then the result is additionally encrypted with the ChaCha20 algorithm. In the first stage, a random 256-bit key for AES and a 128-bit initialisation vector are generated. The data is padded to the required block length and encrypted. The resulting ciphertext is then re-encrypted using ChaCha20 with its own 256-bit key and a nonce (a unique number that does not repeat). During the decryption phase, the reverse sequence of actions is applied: ChaCha20 first removes the outer encryption layer, and then AES-256 decrypts the data, removing the padding to recover the original message. The programme demonstrates the step-by-step transformation of the input message: first, the original text is displayed, followed by its encrypted version after applying AES-256, then the result after additional ChaCha20 encryption. After the decryption steps, the AES-encrypted block is first restored, and finally, the original message is fully recovered (Fig. 3).

```
📝 Original message: b'Confidential data for hybrid symmetric encryption'
🔒 [AES-256] Encrypted message: b'\xfcw\xbeq\xfd\xa8y\x1a]%\t\xa7\xbb\xbek\xea\xb48\xd7\xf
aV\xce\x19\xa3n\xf7\xf2\xae\xe9\xc8C\x00jg\rO\xb5\r\xf6\xe1\xca\xda\xff\xf4\xb0\x95\x9ew\x
18b\xcb\xa8m\x9aQ\\\xf8;wf3\x82ju'
🌀 [ChaCha20] Encrypted message: b'\xea~\x1b\x9fp=\xeb\x86\xc7\n s\x9f\xcbR\xa1D\x9doE\xd0
\xc2\x90]\xd8(\xb0\xe8\xe5\xdf\x99c8\xb0B&u\xbbg`\xe0Y\xfcWY\xe4 v\xb4\x12\x12\x8f)\xdf\xe
c\x8b\xd6\xfe\xdd\xeaR\x88\xf8\x07'
🧩 [ChaCha20] Decrypted (AES encrypted message): b'\xfcw\xbeq\xfd\xa8y\x1a]%\t\xa7\xbb\xbe
k\xea\xb48\xd7\xfaV\xce\x19\xa3n\xf7\xf2\xae\xe9\xc8C\x00jg\rO\xb5\r\xf6\xe1\xca\xda\xff\x
f4\xb0\x95\x9ew\x18b\xcb\xa8m\x9aQ\\\xf8;wf3\x82ju'
✅ [AES-256] Final decrypted message: b'Confidential data for hybrid symmetric encryption'
```

Figure 3. Result of executing the hybrid symmetric encryption and decryption program
Source: created by the author

The obtained result confirms that the encryption and decryption of data using the combined symmetric approach was performed correctly: the encryption structure was preserved, and the output data were fully identical to the input message. Compared to traditional symmetric algorithms, such as the standalone use of AES or ChaCha20, the hybrid encryption method demonstrates a balanced compromise between security and performance. The use of AES in Cipher Block Chaining mode ensures high cryptographic strength, particularly due to standard support for hardware acceleration on modern mobile processors, which positively impacts encryption speed and energy consumption. In contrast, the additional application of ChaCha20, which is optimised for software implementations on resource-constrained devices, enhances resistance to attacks while only slightly increasing computational costs. This hybrid combination is especially relevant for mobile applications, where it is necessary to ensure both high data security and efficient use of device resources. Compared to the standalone use of either AES or ChaCha20, the hybrid algorithm may slightly lag in performance but provides more comprehensive protection, reducing the risk of potential vulnerabilities in individual methods. Thus, it is a promising solution for mobile platforms with heterogeneous hardware support for cryptography and varying performance and security requirements.

On the other hand, asymmetric algorithms such as RSA and ECC are important components of modern hybrid encryption, especially in the context of mobile applications. RSA is based on the mathematical complexity of factoring large prime numbers, and is most commonly used for the secure transmission of symmetric keys. At the same time, ECC, which is based on the algebra of elliptic curves, provides equivalent cryptographic strength with significantly shorter key lengths, making it more energy-efficient and suitable for mobile platforms. Both approaches can be integrated into hybrid cryptographic schemes to combine the strengths – the high speed of symmetric algorithms with secure key exchange through asymmetric methods. Similar to symmetric methods, these algorithms can also be implemented in Python using the cryptography library (Fig. 4).

```
29  def rsa_decrypt_key(private_key, encrypted_key):
30      decrypted_key = private_key.decrypt(
31          encrypted_key,
32          padding.OAEP(
33              mgf=padding.MGF1(algorithm=hashes.SHA256()),
34              algorithm=hashes.SHA256(),
35              label=None
36          )
37      )
38      return decrypted_key
39
40  def generate_ecc_keys():
41      private_key = x25519.X25519PrivateKey.generate()
42      public_key = private_key.public_key()
43      return private_key, public_key
44
45  def compute_shared_secret(private_key, peer_public_key):
46      shared_secret = private_key.exchange(peer_public_key)
47      derived_key = HKDF(
48          algorithm=hashes.SHA256(),
49          length=32,
50          salt=None,
51          info=b'handshake data',
52          backend=default_backend()
```

Figure 4. Results of executing the hybrid asymmetric encryption and decryption code
Source: created by the author

This code demonstrates two key mechanisms of hybrid asymmetric encryption: asymmetric encryption of a symmetric key using RSA and key exchange via ECC. First, a random symmetric key (32 bytes) is generated, which is then encrypted using the RSA public key with the Optimal Asymmetric Encryption Padding algorithm, based on SHA-256, ensuring enhanced resistance to attacks. Decryption is then performed using the private key, followed by a verification of the original and recovered keys, confirming the correctness of the encryption. Next, key pair generation for two parties (A and B) is demonstrated using the X25519 algorithm. Each party uses its private key and the other party's public key to compute a shared symmetric secret through the Elliptic Curve Diffie-Hellman protocol. The results show that both parties obtained identical keys, indicating a successful and secure exchange (Fig. 5).

This approach allows a shared secret to be established without transmitting the key itself, which significantly increases resistance to interception and manipulation in open environments. Moreover, the hybrid encryption method combining RSA and ECC has substantial advantages compared to using each of these methods individually. The traditional use of RSA alone for data protection provides a reliable level of security but is characterised by relatively large

key sizes and slower performance, which is especially noticeable on mobile devices with limited resources. In turn, using ECC alone offers high cryptographic strength with significantly shorter key lengths and better performance; however, its standalone use may complicate compatibility with existing protocols and may require additional configuration. The current approach combines the strengths of both algorithms: RSA effectively protects the symmetric key, while ECC ensures secure and efficient key exchange, enhancing the overall security and performance of the system. Thus, the hybrid approach optimises the balance between security, processing speed, and resource consumption, making it particularly appropriate for mobile applications and environments with heightened data protection requirements.

Figure 5. Results of executing the hybrid asymmetric encryption and decryption code
Source: created by the author

Analysis of the possibilities of integrating hybrid cryptographic algorithms into mobile applications and assessment of the effectiveness

The integration of hybrid cryptographic algorithms into mobile applications requires consideration of several specific constraints inherent to the mobile environment. The main factors affecting the implementation of cryptographic solutions on mobile devices include: limited hardware resources (Central Processing Unit, Random Access Memory, battery); specific runtime environments, such as ART/Dalvik, iOS runtime; and restrictions related to energy consumption and security maintenance. At the same time, these factors also encompass the availability of advanced hardware security tools (Secure Enclave, Advanced RISC Machine (ARM) Crypto Extensions), which open up new possibilities for cryptographic optimisation. Table 1 provides a comparative overview of the main constraints and capabilities of the mobile environment in the context of implementing hybrid cryptographic algorithms that combine symmetric and asymmetric encryption.

Table 1. Analysis of mobile environment factors for cryptographic algorithms

Factor	Limitation	Opportunities for symmetric algorithms	Opportunities for asymmetric algorithms
Central Processing Unit	Limited processing power of mobile processors requires efficient algorithms	AES: efficient with hardware acceleration (encryption time ~275.6 ms)	RSA: high load due to slow operations (key generation ~2,532.8 ms)
		ChaCha20: faster in software (~2 ms)	ECC: faster, lower load (~15.8 ms)
Random Access Memory	Limited RAM (2-8 GB) requires low memory usage	AES: low usage (~17.74 MB)	RSA: moderate memory usage
		ChaCha20: higher (~148.4 MB), needs optimisation	ECC: less usage due to compact keys
Energy consumption	Limited battery capacity requires energy-efficient algorithms	AES: high without hardware acceleration	RSA: high due to complex calculations
		ChaCha20: lower due to speed in software	ECC: lower due to smaller keys and faster operations
Android runtime	ART/Dalvik affects cryptographic operations, limiting performance	Algorithms are supported via Android Security library, AES is more efficient with hardware acceleration	Both algorithms are supported, ECC is faster and more efficient on ARMv8
iOS runtime	iOS runtime with CommonCrypto and Secure Enclave has limitations on implementation flexibility	Supported via CommonCrypto, ES is more efficient with Secure Enclave	Both algorithms are supported, ECC is more efficient due to hardware support
Libraries and Application Programming Interface	Limited support for complex cryptographic libraries in mobile operating systems	Available via Android Security and CommonCrypto, providing flexible integration	Supported through standard libraries, ECC is easier to integrate due to smaller keys
Secure key storage	Need to protect keys from unauthorised access	Use Keystore (Android) and Keychain (iOS) for secure storage of symmetric keys	Use Keystore/Keychain, ECC is more efficient due to smaller key sizes

Notes: ARMv8 – Advanced RISC Machine version 8
Source: created by the author

Thus, symmetric algorithms, particularly AES and ChaCha20, demonstrate high efficiency when hardware acceleration or software execution optimisation is available, whereas asymmetric methods (RSA, ECC) differ significantly in terms of resource load, with ECC being more suitable for mobile environments due to smaller key sizes and faster operations. Given the above characteristics of cryptographic algorithms, it is also important to understand the capabilities provided by modern mobile platforms for the implementation. Different operating systems, such as Android and iOS, offer the own cryptographic frameworks, libraries, and key storage and processing tools, which directly affect the adaptation and optimisation of hybrid solutions.

The Android platform provides developers with a wide range of cryptographic tools, particularly Android Keystore, which ensures secure key storage, and the Bouncy Castle library, which supports various cryptographic algorithms. The Spongy Castle and Conscrypt libraries are also important. These tools enable effective implementation of both symmetric and asymmetric encryption algorithms. In particular, the use of hybrid cryptographic models, such as the combination of ECC and Blowfish, in conjunction with deep learning, improves the accuracy of malware detection on Android devices. For example, a model combining deep learning (AlexNet classifier) with hybrid cryptography (ECC and Blowfish) for detecting ransomware on Android achieves a detection accuracy of 99.89%, outperforming traditional methods such as Graph Neural Network (94.76%), Convolutional Neural Network (95.76%), and Random Forest (96%) (Kalphana *et al*. 2024).

On the other hand, iOS offers a set of cryptographic frameworks, notably CryptoKit and CommonCrypto, which provide access to modern encryption, hashing, and digital signature algorithms. To enhance security, iOS also actively uses hardware means, such as the Secure Enclave, which securely stores keys in isolation and performs critical cryptographic operations, and OpenSSL, which offers flexible support for a broad range of cryptographic algorithms. For example, the comprehensive approach of iOS to data security includes protection of data "at rest", during transmission, and secure communication via SSL/TLS protocols (Prodduturi, 2025). The comparison of different mobile platforms was illustrated in Table 2.

Table 2. Comparative analysis of mobile platforms

Tool	Platform	Algorithm support	Features and limitations	Opportunities for hybrid and post-quantum algorithms
Keystore	Android	AES, RSA, ECC, partially Kyber, Dilithium	Hardware-secured key storage, limited support for ChaCha20 and post-quantum algorithms	Secure key storage for AES, RSA, ECC; post-quantum algorithms require external libraries
Conscrypt			Optimised for Android, integrated with ART, limited support for ChaCha20 and SPHINCS+	Fast for AES, ECC with ARMv8 Crypto Extensions; post-quantum algorithms require additional libraries
Bouncy Castle		AES, ChaCha20, RSA, ECC, Kyber, Dilithium, Falcon, SPHINCS+	Flexible, but resource-intensive, needs optimisation for mobile devices	Full support for hybrid (AES+ChaCha20, RSA+ECC) and post-quantum algorithms, but slower without hardware acceleration
Spongy Castle			A lighter version of Bouncy Castle, adapted for Android, but less active support	Efficient for hybrid and post-quantum algorithms on older devices, supports all algorithms
CryptoKit	iOS	AES, ChaCha20, ECC, partially Kyber	Modern, iOS-optimised, limited support for RSA and post-quantum algorithms	Suitable for hybrid schemes (ChaCha20+ECC), partial support for Kyber, requires OpenSSL for other post-quantum algorithms
Common-Crypto		AES, ChaCha20, RSA, ECC	Reliable but less flexible, no direct support for post-quantum algorithms	Full support for hybrid schemes, but post-quantum algorithms require external libraries
OpenSSL		AES, ChaCha20, RSA, ECC, Kyber, Dilithium, Falcon, SPHINCS+	Flexible but complex integration due to the closed iOS ecosystem, high resource intensity	Supports all hybrid and post-quantum algorithms, but needs optimisation for iOS

Notes: SPHINCS – Stateless Practical Hash-based Incredibly Nice Crypto Signature
Source: created by the author

Overall, Keystore is the standard Android mechanism for securely storing cryptographic keys in an encrypted, hardware-backed container, ensuring a high level of data protection. An addition to this is Conscrypt – a cryptographic provider from Google, specifically optimised for working with TLS and integrated into the ART runtime environment, which contributes to the efficient execution of cryptographic operations on Android devices. For extended algorithm support, Bouncy Castle was used – a universal cryptographic library offering a wide range of symmetric, asymmetric, and post-quantum algorithms, including Kyber, Dilithium, and SPHINCS+. For resource-constrained devices, Spongy Castle is available – a lightweight and Android-adapted version of Bouncy Castle that ensures compatibility with legacy devices while maintaining support for core algorithms.

On the iOS platform, CryptoKit plays an important role – a modern framework that supports ChaCha20, ECC, and is closely integrated with the Secure Enclave hardware module, enhancing the security of cryptographic operations. Also in use is CommonCrypto – a low-level C-library from Apple, which focused on implementing basic cryptographic functions with support for AES, RSA, and ECC, which provides a stable though less flexible implementation. No less important is OpenSSL – a powerful and flexible library supporting all major and post-quantum algorithms, including Falcon and Dilithium; however, its integration into iOS is often complicated by the closed nature of Apple's ecosystem. Thus, these tools create a broad field for selecting optimal cryptographic components when building hybrid security models, allowing the adaptation of algorithms to the specifics of a given mobile platform.

Therefore, on Android, the hybrid method ChaCha20 + AES provides fast encryption (~2 ms for ChaCha20, ~275.6 ms for AES with hardware acceleration), whereas RSA + ECC optimises key exchange (~15.8 ms for ECC versus ~2,532.8 ms for RSA), which is significantly faster than using RSA alone. On iOS, thanks to CryptoKit and Secure Enclave, ChaCha20 + AES achieves comparable encryption speed, and ECC reduces load compared to RSA, ensuring better energy efficiency. The separate use of AES or RSA is slower and less energy-efficient, especially without hardware acceleration, whereas the hybrid approach optimises performance and security, particularly through the use of Keystore (Android) and Keychain (iOS) for key protection. Moreover, hybrid algorithms reduce execution latency and resource consumption, which is critical for mobile devices. For instance, in the hybrid approach, ChaCha20 reduces reliance on hardware acceleration, unlike AES, and ECC provides faster, and more compact key exchange compared to RSA. Although individual algorithms like ChaCha20 (~148.4 MB memory usage) may impose greater memory load than AES (~17.74 MB), the hybrid method compensates for this through comprehensive security and adaptability to various hardware configurations, making it more effective for mobile applications.

DISCUSSION

The results of this study highlighted that hybrid cryptographic algorithms provide effective protection of personal data in mobile applications due to the combination of symmetric and asymmetric encryption and optimisation for the mobile environment. In turn, the study by V.B. Bhimanapati *et al.* (2024) applied artificial intelligence and machine learning to detect vulnerabilities and behavioural threats in mobile applications. Thus, both studies align: the present one – in the field of cryptographic protection, and the mentioned one – in the aspect of dynamic monitoring and adaptive security.

The results showed that the proposed hybrid cryptographic algorithms not only enhance the security of personal data but are also effectively adapted to the limited resources of mobile devices, ensuring performance, energy efficiency, and low memory usage. At the same time, the study by A. Grace (2025) focused on adaptive solutions for mobile security, which apply machine learning and behavioural analytics to detect threats in real time. Both approaches complement each other: the current study strengthens protection through cryptographic methods at the data level, while the cited study was aimed at active threat detection and prevention at the level of user behaviour and system activity.

The current conclusions also demonstrated that the integration of symmetric (AES, ChaCha20) and asymmetric (RSA, ECC) algorithms within hybrid cryptography allows achieving a balance between high security and performance, which is especially important for mobile applications with limited resources. Meanwhile, in the study by V.R. Kanagavalli & A. Meenakshi (2024), the main focus was placed on the theoretical analysis of the interrelation between machine learning and cryptography, in particular – how machine learning can improve cryptographic methods, or how encryption can be used to protect models and machine learning data. Both studies demonstrated the potential for interdisciplinary integration, albeit from different perspectives: the current research focused on the implementation of hybrid algorithms in mobile environments, whereas the mentioned study offered a conceptual generalisation of the interaction possibilities between machine learning and cryptography.

In addition, in the conducted study, AES-256 was implemented as a key element of hybrid symmetric encryption, providing high cryptographic strength, efficiency due to hardware acceleration, and adaptation to mobile environments with limited resources. Similarly, in the study by W. Abbas *et al.* (2025), AES was applied for message encryption, ensuring fast and reliable protection of personal messages in mobile networks, compatible with most Android devices. The results of both studies align, confirming the high efficiency of AES in mobile environments as a universal tool for the protection of confidential data.

The conclusions of this study emphasised that the implementation of hybrid cryptographic algorithms, particularly using AES, ChaCha20, RSA, and ECC, not only increases the level of protection of personal data in mobile applications but also takes into account the requirements for data confidentiality and security in the modern digital environment. Similarly, in the study by P.M. Bandara *et al.* (2025), the importance of cryptographic technologies such as encryption and anonymisation for complying with international data protection standards, particularly the General Data Protection Regulation (2016) and the California Consumer Privacy Act (2018), was highlighted. Both studies complement each other, showing that cryptographic solutions can serve as both a technical means of data

protection and a tool for ensuring compliance with privacy legislation.

The ChaCha20 algorithm analysed in this study is a high-performance stream cipher optimised for mobile applications with limited resources, where it, as part of a hybrid scheme with AES-256, provides a balanced compromise between security and performance, in particular due to its fast software implementation and moderate energy consumption, although it requires some memory optimisation. In the study by F.A. Fadhil *et al*. (2024), ChaCha20 was applied for encrypting data before hiding these data in video frames using the Laplacian of Gaussian algorithm, which demonstrated high speed, reliable protection, and preservation of video quality, confirmed by experimental metrics. Therefore, the results of the current study confirmed the conclusions of the mentioned study, demonstrating the versatility and efficiency of ChaCha20 for both mobile cryptography and multimedia information protection.

Regarding the RSA algorithm, the current results indicated that it provides a high level of security but involves significant computational costs and slow key generation, which limits its efficiency on mobile devices. Similarly, the authors Y. Gao *et al*. (2023), in the blockchain model, applied RSA for data encryption and digital signing, noting its high cryptographic strength but also emphasising the need for optimisation due to high computational load. This coincides with the current conclusions regarding the limitations of RSA in the mobile environment.

The results of this study generally align with the findings presented in the work by Z. Chen *et al*. (2023), which proposed a hybrid cryptographic algorithm Hybrid AES and ECC, combining symmetric data encryption using AES and asymmetric key encryption using ECC. As in the current study, the authors emphasised the efficiency of the hybrid approach in enhancing security, optimising key management, and ensuring a balance between performance and protection. Both studies demonstrated the advantages of hybrid cryptography under resource constraints: in this case – mobile applications, in the mentioned study – blockchain systems.

The conducted study emphasised that ECC provides high cryptographic strength, fast key generation (~15.8 ms), and low energy consumption in mobile applications. Moreover, in the study by Y. Yan (2022), ECC was noted as one of the strongest and most efficient modern cryptographies, capable of optimisation through hardware acceleration. Thus, both studies agree in the conclusion that ECC is the optimal choice for secure and efficient encryption under limited mobile resources. While the current work focused on the implementation of hybrid algorithms for the mobile environment, including RSA and ECC, the study by C. Gitonga (2025) analysed the risks for these algorithms in the context of quantum attacks and considered post-quantum alternatives such as CRYSTALS-Kyber and SPHINCS+. That is, the current work has an advantage in the applied aspect, as it offers adapted solutions for modern mobile platforms.

Whereas the present study focused on the implementation of hybrid cryptographic algorithms for the protection of personal data in mobile applications, the work by F. Ogwara *et al*. (2025) concentrated on the application of machine learning and stochastic methods for threat detection in mobile cloud computing environments. Both approaches are complementary: the current work proposed universal solutions for data encryption and storage, while the other study focused on dynamic monitoring and anomaly detection. The advantage of the current approach lies in its lower implementation complexity and compatibility with a wide range of mobile platforms without the need for cloud infrastructure.

In discussing the effectiveness of hybrid cryptographic schemes for protecting personal data in mobile applications, it is instructive to reference V.V. Lukichov *et al*. (2023), who proposed an adaptive multi-layer protection method combining steganography and cryptography. Their approach demonstrates how these two techniques can synergistically enhance resistance to attacks, assessed using metrics such as MSE and PSNR. Compared to their work, the proposed hybrid scheme similarly integrates symmetric encryption with asymmetric authentication mechanisms (e.g. AES with RSA or ECC), aiming to achieve a balance of confidentiality and integrity tailored to mobile environments. Additionally, the study by V.I. Malinovskyi *et al*. (2024) presented a mathematical model for evaluating cyber-threats and informational impacts within microcontroller systems. Their analysis of specific risk vectors and threat dynamics provides a useful parallel to the mobile application context described in this study, where separation of key management (via ECC/RSA) and symmetric encryption modules enhances both scalability and robustness.

In general, this study implemented a hybrid symmetric encryption approach combining AES-256 and ChaCha20, which enabled a balance between performance and security on mobile devices. Compared to the results of M. Vinothkumar & S. Ram (2025), who proposed a modified AES with chaotic key generation to enhance the security of electronic documentation, the current implementation demonstrates lower computational costs and better suitability for the mobile environment due to AES hardware acceleration support and the software efficiency of ChaCha20. Although the mentioned approach strengthens the dynamic nature of keys, it requires more complex integration into mobile platforms. In turn, compared to the study by W.B. Nugroho *et al*. (2024), in which ChaCha20 was successfully applied for image encryption, the current approach was aimed not only at images but at universal encryption of textual data in mobile applications, providing additional resistance through cascading encryption and key storage via Keystore/Keychain.

The conclusions of this study regarding RSA (key generation ~2,532.8 ms) confirmed that not only its high reliability but also its limitations in mobile

environments due to significant computational costs, which aligns with the findings of J.C. Vimala (2023), who emphasised its advantages in cloud security but acknowledged implementation complexity. In turn, R. Alsowail (2025) demonstrated the efficiency of ECC in video encryption with an encryption time of 4.49 seconds and security of 98.5%. At the same time, the current approach using ECC (Central Processing Unit ~15.8 ms) confirmed its speed and resource efficiency for mobile applications, making the combined use of RSA and ECC an optimal solution.

Additionally, the current conclusions regarding the hybrid use of cryptographic algorithms are consistent with the results of the study by Z. Yasqi *et al*. (2025), which confirmed the efficiency of combining symmetric (ChaCha20) and asymmetric (RSA) algorithms to ensure reliable data protection against various cyber threats. Both studies note successful encryption and decryption operation, as well as confirm the system's resistance to unauthorised access. Meanwhile, the present study focuses on performance optimisation and energy efficiency in the mobile environment, which is a logical continuation of the directions proposed by the authors for further development – improving algorithm efficiency and reducing execution time.

Thus, the results of the study confirmed that hybrid cryptographic algorithms are an effective and practical solution for the protection of personal data in mobile applications, ensuring a balance between security, performance, and resource efficiency. The proposed approaches were adapted to the specifics of the mobile environment and meet modern requirements for confidentiality and information protection. The implementation of such solutions will contribute to raising the overall level of cybersecurity in mobile technologies.

CONCLUSIONS

At the first stage of the study, a concept of hybrid encryption was proposed, combining symmetric (AES, ChaCha20) and asymmetric (RSA, ECC) cryptographic algorithms with the aim of achieving a balance between performance, security, and efficient use of mobile device resources. The results demonstrated the practical implementation of two variants of the hybrid approach in Python: symmetric double encryption using AES-256 and ChaCha20, as well as a scheme for secure transmission of a symmetric key using RSA and key exchange based on ECC (X25519). The obtained results emphasised the correctness of both approaches, confirming the feasibility for mobile applications, where limited computational resources require optimisation of cryptographic solutions without compromising the level of personal data protection.

The second stage of the study showed that effective integration of hybrid cryptographic algorithms into mobile applications is only possible with a deep understanding of the mobile environment's specifics. For example, symmetric algorithms (AES, ChaCha20) demonstrated high performance on devices with hardware support – AES achieved encryption times of ~275.6 ms with low memory usage (~17.74 MB), whereas ChaCha20 provided even higher speed (~2 ms), although it consumed more memory (~148.4 MB). In the context of asymmetric algorithms, ECC proved significantly more efficient than RSA: ECC key generation took ~15.8 ms versus ~2532.8 ms for RSA, which also positively affected power consumption and overall system load. Accordingly, hybrid models enable an optimal balance between speed, security, and resource consumption. Platform-specific features, such as the use of Android Keystore or iOS Keychain, support for Secure Enclave, ARM Crypto Extensions, further enhance these advantages, especially on modern devices. Analysis of such tools confirmed that Android provides broad support for both classical and post-quantum algorithms (via Bouncy Castle, Spongy Castle, Conscrypt), while iOS offers a more tightly controlled but hardware-optimised environment (CryptoKit, CommonCrypto). Thus, the proposed hybrid approach, combining symmetric encryption ChaCha20 or AES with asymmetric ECC or RSA, demonstrated better efficiency than traditional algorithms in terms of both speed and energy efficiency.

A limiting aspect of this study is that testing was carried out on a limited number of devices and emulators, which does not allow for a full coverage of the variability of mobile platforms and scenarios of the practical use. Also, performance analysis of the algorithms was not conducted in the context of different hardware configurations, particularly with regard to long-term load and changes in power consumption under real operating conditions. In this regard, it is advisable in further studies to expand the experimental base through testing on various devices and long-term monitoring of resource consumption, as well as to analyse the possibilities of adaptive integration of hybrid and post-quantum cryptographic algorithms depending on the dynamics of the environment and security threats.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abbas, W., Joshua, S.R., Abbas, A., & Lee, J.-H. (2025). An end-to-end GSM/SMS encrypted approach for smartphone employing advanced encryption standard (AES). *ArXiv*. doi: 10.48550/arXiv.2503.18859.

[2] Ali, A.K., Raza, A., & Arif, H. (2025). AI-driven dynamic selection of post-quantum algorithms for mobile application security. *Asian Bulletin of Big Data Management*, 5(2), 51-62. doi: 10.62019/eq34ar93.
[3] Alsowail, R. (2025). Advanced video encryption using the opposition lotus effect-elliptic curve cryptography in signal processing applications. *Signal Image and Video Processing*, 19(5), article number 409. doi: 10.1007/s11760-025-03899-x.
[4] Bandara, P.M., Abeyrathne, P., & Sandirigama, M. (2025). Cryptographic technologies for guaranteeing compliance of data privacy to international data protection laws – a preliminary study. In D. Dahanayake & M. Rabindrakumar (Eds.), *Transformative applied research in computing, engineering, science and technology* (pp. 237-243). London: CRC Press. doi: 10.1201/9781003616368-32.
[5] Bhimanapati, V.B., Jain, S., & Pandian, P.K. (2024). Security testing for mobile applications using AI and ML algorithms. *Journal of Quantum Science and Technology*, 1(2), 44-58. doi: 10.36676/jqst.v1.i2.15.
[6] Borysenko, O., & Tymoshenko, A. (2024). Overview of personal data protection methods in the cloud environment. *Infocommunication and Computer Technologies*, 1(7), 31-34. doi: 10.36994/2788-5518-2024-01-07-04.
[7] California Consumer Privacy Act. (2018, June). Retrieved from https://oag.ca.gov/privacy/ccpa.
[8] Chen, Z., Gu, J., & Yan, H. (2023). HAE: A hybrid cryptographic algorithm for blockchain medical scenario applications. *Applied Sciences*, 13(22), article number 12163. doi: 10.3390/app132212163.
[9] Fadhil, F.A., Tawfiq, F., & Thamer, M. (2024). Enhancing data security using laplacian of gaussian and Chacha20 encryption algorithm. *Journal of Intelligent Systems*, 33(1), article number 20240191. doi: 10.1515/jisys-2024-0191.
[10] Fan, H., Meng, L., Zheng, F., Mingyu, W., & Bowen, X. (2022). Black-box testing of cryptographic algorithms based on data characteristics. In J. Lin & Q. Tang (Eds.), *Applied cryptography in computer and communications* (pp. 153-169). Cham: Springer. doi: 10.1007/978-3-031-17081-2_10.
[11] Gao, Y., Guo, L., & Zhang, T. (2023). Exploring and envisioning the application of blockchain technology for privacy data protection. *Applied and Computational Engineering*, 19(1), 123-131. doi: 10.54254/2755-2721/19/20231020.
[12] General Data Protection Regulation. (2016, May). Retrieved from https://gdpr-info.eu/.
[13] Gitonga, C. (2025). The impact of quantum computing on cryptographic systems: Urgency of quantum-resistant algorithms and practical applications in cryptography. *European Journal of Information Technologies and Computer Science*, 5(1), 1-10. doi: 10.24018/compute.2025.5.1.146.
[14] Gour, A., Malhi, S.S., Singh, G., & Kaur, G. (2024). Hybrid cryptographic approach: For secure data communication using block cipher techniques. *E3S Web of Conferences*, 556, article number 01048. doi: 10.1051/e3sconf/202455601048.
[15] Grace, A. (2025). *Advancements in mobile device security: Developing AI-powered applications for enhanced protection*. Retrieved from https://www.researchgate.net/publication/389173473_Advancements_in_Mobile_Device_Security_Developing_AI-Powered_Applications_for_Enhanced_Protection.
[16] Kalphana, K.R., Aanjankumar, S., Surya, M., Ramadevi, M.S., Ramela, K.R., Anitha, T., Nagaraj, N., & Ramaswamy, K. (2024). Prediction of android ransomware with deep learning model using hybrid cryptography. *Scientific Reports*, 14(1), article number 22351. doi: 10.1038/s41598-024-70544-x.
[17] Kanagavalli, V.R., & Meenakshi, A. (2024). A survey of cryptographic data protection and machine learning. In J.A. Ruth, V.G. Mahesh, P. Visalakshi, R. Uma & A. Meenakshi (Eds.), *Machine learning and cryptographic solutions for data protection and network security* (pp. 1-11). London: IGI Global. doi: 10.4018/979-8-3693-4159-9.ch001.
[18] Khalid, R., Najat, Z., & Sayda, S.J. (2025). A hybrid approach to cloud data security using ChaCha20 and ECDH for secure encryption and key exchange. *Kurdistan Journal of Applied Research*, 10(1), 66-82. doi: 10.24017/science.2025.1.5.
[19] Lukichov, V.V., Baryshev, Y.V., Kondratenko, N.R., & Malinovskyi, V.I. (2023). Adaptive multi-layer information protection method combining steganography and cryptography. *Information technology and computer engineering*, 3, 4-11. doi: 10.31649/1999-9941-2023-58-3-4-11.
[20] Malinovskyi, V.I., Kupershtein, L.M., & Lukichov, V.V. (2024). A mathematical model for assessing cyber-threats and informational impacts in microcontrollers. *Information technology and computer engineering*, 59(1), 69-82. doi: 10.31649/1999-9941-2024-59-1-69-82.
[21] Muthaura, A., & Kandiri, J. (2024). Data protection in healthcare information systems using cryptographic algorithm with Base64 512 bits. *Open Journal for Information Technology*, 7(1), 11-22. doi: 10.32591/coas.ojit.0701.02011m.
[22] Neve, R., & Bansode, R. (2024). Attack analysis on hybrid-SIMON-SPECKey lightweight cryptographic algorithm for IoT applications. *Indian Journal of Science and Technology*, 17(10), 932-940. doi: 10.17485/IJST/v17i10.2811.
[23] Nugroho, W.B., Susanto, A., Sari, A., Rachmawanto, E.H., & Doheir, M. (2024). A robust and imperceptible for digital image encryption using Chacha20. *Jurnal Teknik Informatika*, 5(2), 397-404. doi: 10.52436/1.jutif.2024.5.2.1470.

[24] Nwatuzie, G.A., Ijiga, O.M., Idoko, I.P., Enyejo, L.A., & Ali, E.O. (2025). Design and evaluation of a user-centric cryptographic model leveraging hybrid algorithms for secure cloud storage and data integrity. *American Journal of Innovation in Science and Engineering*, 4(2), 49-65. doi: 10.54536/ajise.v4i2.4482.
[25] Ogwara, F., Petrova, K., Yang, M.L., & MacDonell, S.G. (2025). Mindpres: A hybrid prototype system for comprehensive data protection in the user layer of the mobile cloud. *Sensors*, 25(3), article number 670. doi: 10.3390/s25030670.
[26] Pitale, R.R., Tajane, K.D., Mahajan, P.B., Nehate, N.S., Mulimani, A.J., & Lokhande, D.S. (2024). Cryptographic algorithm development and application for encryption and decryption. In D. Goyal, A. Kumar, D. Singh, M. Paprzycki, P. Jain, B.B. Gupta & U.P. Singh (Eds.), *ICIMMI '23: Proceedings of the 5th international conference on information management & machine intelligence* (article number 27). New York: Association for Computing Machinery. doi: 10.1145/3647444.3647853.
[27] Prodduturi, S.M. (2025). Cryptography in iOS: A study of secure data storage and communication techniques. *International Journal on Science and Technology*, 16(1). doi: 10.71097/IJSAT.v16.i1.1403.
[28] Sayed, M.A. (2024). A comparative study of machine learning-based and traditional cryptographic methods for personal data encryption. *ResearchGate*. doi: 10.13140/RG.2.2.33661.88803.
[29] Semerenska, V. (2025). Quantum-resistant cryptographic algorithms for critical infrastructures. *Visnyk of Kherson National Technical University*, 2(1), 204-209. doi: 10.35546/kntu2078-4481.2025.1.2.27.
[30] Sereda, A., Datsenko, I., Pavlenko, V., & Samarai, V. (2023). Stability and efficiency of cryptographic algorithms used in mobile devices. *Infocommunication and Computer Technologies*, 2(4), 178-190. doi: 10.36994/2788-5518-2022-02-04-21.
[31] Vimala, J.C. (2023). Securing cloud environments: A comparative analysis of RSA and MRGA for enhanced data protection. *Journal on Cloud Computing*, 10(1), 38-46. doi: 10.26634/jcc.10.1.19937.
[32] Vinothkumar, M., & Ram, S. (2025). Blockchain-based modified AES with chaotic random key generation for secured E-medical data sharing. *CLEI Electronic Journal*, 28(2). doi: 10.19153/cleiej.28.2.14.
[33] Yan, Y. (2022). The overview of elliptic curve cryptography (ECC). *Journal of Physics Conference Series*, 2386, article number 012019. doi: 10.1088/1742-6596/2386/1/012019.
[34] Yasqi, Z., Hayaty, N., & Bettiza, M. (2025). Cryptography of Chacha20 and RSA algorithms for text security. *Journal of Computer Networks Architecture and High Performance Computing*, 7(1), 290-301. doi: 10.47709/cnahpc.v7i1.5345.

Розробка та тестування ефективності гібридних криптографічних алгоритмів для захисту персональних даних у мобільних додатках

Віталій Ясененко

Магістр, старший розробник програмного забезпечення
TP-Link
92618, вул. Технологій, 36, м. Ірвайн, США
https://orcid.org/0009-0004-4801-9541

Анотація. Зі стрімким зростанням кількості мобільних додатків та обсягу оброблюваних персональних даних підвищується потреба у впровадженні ефективних засобів їхнього захисту. Мета роботи полягала в розробці гібридних криптографічних алгоритмів і в аналізі можливостей їх інтеграції в мобільні додатки для підвищення рівня захисту персональних даних. У дослідженні реалізовано гібридні криптографічні алгоритми, які поєднують симетричні та асиметричні методи шифрування, проведено моделювання їхньої роботи на прикладі програмних модулів мовою Python, а також здійснено оцінку їх ефективності з урахуванням характеристик мобільного середовища. Основні результати показали, що поєднання алгоритмів симетричного шифрування Advanced Encryption Standard і ChaCha20 забезпечує повне збереження даних при дешифруванні та дозволяє зменшити ризики компрометації завдяки двошаровій структурі шифрування. Також встановлено, що під час тестування асиметричних методів алгоритм Rivest-Shamir-Adleman успішно захищає симетричні ключі, а алгоритм Elliptic Curve Cryptography забезпечує обчислення спільного секрету двома сторонами без передачі самого ключа, що підвищує стійкість до перехоплення. Результати програмної реалізації підтвердили ідентичність вхідних і вихідних даних, що засвідчує надійність гібридного підходу до шифрування. З іншого боку, на Android платформах Keystore, Bouncy Castle, Spongy Castle і Conscrypt забезпечують швидке шифрування Advanced Encryption Standard (~275,6 мілісекунд з апаратним прискоренням) і ChaCha20 (~2 мілісекунди), а також ефективний обмін ключами через Elliptic Curve Cryptography (~15,8 мілісекунд) порівняно з Rivest-Shamir-Adleman (~2532,8 мілісекунд), з підтримкою постквантових алгоритмів. На iPhone Operating System платформах CryptoKit, CommonCrypto і Open Secure Sockets Layer забезпечують схожу швидкість шифрування, а Secure Enclave оптимізує Elliptic Curve Cryptography, хоча постквантові алгоритми потребують додаткової оптимізації. Гібридний підхід знижує використання пам'яті, що робить алгоритми ефективними для мобільних пристроїв. Отримані результати можуть використовувати розробники мобільних додатків для покращення захисту даних у фінансових, медичних та корпоративних системах на смартфонах з операційними системами Android і iPhone

Ключові слова: розширений стандарт шифрування; ChaCha20; Rivest-Shamir-Adleman; криптографія на основі еліптичних кривих; Android; операційна система iPhone

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 621.395.7
DOI: 10.62660/bcstu/3.2025.37

A method for detecting DDoS attacks in VoIP systems based on machine learning

Viktor Gnatyuk*
PhD in Technical Sciences, Associate Professor
State University "Kyiv Aviation Institute"
03058, 1 Lyubomyra Huzara Ave., Kyiv, Ukraine
The State Scientific and Research Institute of Cybersecurity Technologies and Information Protection
03142, 3 Maksyma Zaliznyaka Str., Kyiv, Ukraine
https://orcid.org/0000-0002-4916-7149

Ivan Gorbachov
Postgraduate Student
State University "Kyiv Aviation Institute"
03058, 1 Lyubomyra Huzara Ave., Kyiv, Ukraine
https://orcid.org/0009-0002-9688-1692

Abstract. Protecting VoIP systems from DDoS attacks is a critical issue, as such attacks can lead to significant financial losses and a decline in the quality of service for users. Existing methods of detecting attacks are based on signature analysis or traditional rules, which limits their effectiveness in cases of new or modified attacks. The aim of this work was to develop a method for detecting DDoS attacks in VoIP systems based on machine learning, which provides high accuracy in classifying abnormal traffic. To achieve this goal, methods of network traffic analysis, machine learning, and statistical evaluation of model effectiveness were used. The main research tool was a multilayer perceptron neural network trained on real network traffic. As a result of this research, a model was developed and tested that demonstrated high accuracy in detecting attacks. A comparative analysis of the effectiveness of the developed model with other approaches was carried out. The proposed method was integrated into the Asterisk environment through the Asterisk Manager Interface, which made it possible to monitor SIP traffic in real time, analyse it using a trained model, and automatically block attacking IP addresses through IPTables or Fail2Ban. Based on the results of the model comparison by metrics, the best model was selected and an algorithm for protecting VoIP from DDoS was developed based on it. The practical value of the work lies in the development of an effective method for protecting VoIP systems, which can be used to improve the level of security in telecommunications networks. The proposed approach can be scaled and adapted to different network infrastructure configurations

Keywords: IP telephony; SIP; Asterisk; neural network; cybersecurity

INTRODUCTION

Distributed Denial of Service (DDoS) attacks have become one of the most pervasive threats to modern networks, particularly in real-time communication systems such as Voice over IP (VoIP) networks. These attacks aim to overwhelm the target system by flooding it with massive amounts of traffic, causing service disruptions, degraded quality, or complete downtime. The growing reliance on VoIP for both personal and business

Article's History: Received: 24.04.2025; Revised: 05.08.2025; Accepted: 15.09.2025.

Suggested Citation:
Gnatyuk, V., & Gorbachov, I. (2025). A method for detecting DDoS attacks in VoIP systems based on machine learning. *Bulletin of Cherkasy State Technological University*, 30(3), 37-46. doi: 10.62660/bcstu/3.2025.37.

*Corresponding author

communication makes DDoS attacks particularly disruptive, as they not only affect the availability of the service but also the quality of communication. Consequently, DDoS detection and mitigation in VoIP systems are critical research areas within the broader field of network security.

Moreover, the integration of deep learning techniques has further advanced the field. A. Hekmati *et al.* (2023) proposed the use of correlation-aware neural networks for DDoS detection in IoT systems, emphasising the importance of capturing the relationships between different network parameters for more accurate detection. This insight has been directly applied in the current study, where various traffic metrics (e.g., packet loss, jitter, and delay) were incorporated into the machine learning model, thereby improving the ability to detect subtle attack patterns in VoIP traffic.

In addition to machine learning, autoencoders have become an important tool for anomaly detection, as highlighted by D. Ilin & I. Starinskyi (2023). Autoencoders are unsupervised learning models that can reconstruct input data and detect anomalies by identifying deviations from the reconstructed patterns. This technique is particularly effective in situations where the attack patterns are not well-defined or previously seen. By incorporating autoencoders into detection model presented in this research, authors plan to improve the detection of new, previously unseen DDoS attacks that do not match typical attack signatures. The work of M. Chornobuk *et al.* (2023) explored the problem of detecting DDoS attacks – specifically SYN flood, ICMP flood, and UDP flood – by reviewing existing machine learning-based methods (neural networks, SVM, decision trees). The researchers proposed their own decision tree-based model, trained it on a public dataset of 104,345 records and 23 features (IP address, port, bytes, etc.), rejected features that could cause overfitting, and performed 5-fold stratified cross-validation; the model demonstrated an average classification accuracy of 0.94 ($\sigma = 0.06$), demonstrating its ability to effectively identify malicious traffic in conditions close to real-world conditions, with the possibility of practical application on network equipment for filtering or alerts.

The application of hybrid deep learning models has proven to be particularly effective in the detection of flooding DDoS attacks, as demonstrated by B. Habib & F. Khurshid (2024). The inclusion of both Long Short-Term Memory (LSTM) networks, which capture temporal dependencies, and Convolutional Neural Networks (CNNs), which can identify spatial paterns in network traffic, offers a promising direction for enhancing the robustness of the detection model in future studies. The use of hybrid LSTM-CNN models to detect time-based DDoS attacks presented an innovative approach that can further be adapted to VoIP systems.

In cybersecurity environment, traditional DDoS detection systems are primarily focused on recording attacks that are already active or ongoing, which limits their ability to respond proactively. Some researchers proposed predicting the start of an attack before it occurs. In their work, V.A. Savchenko & B.S. Stepanchenko (2024) developed a model that predicts the expected behaviour trajectory of each connection using an analytical a posteriori approach and, based on this, detects abnormal deviations that may signal slow DDoS attacks (e.g., RUDY). The method includes a linear representation of a random process, trajectory extrapolation, and comparison with threshold values, resulting in the ability to accurately predict the start of an attack before irreversible service degradation occurs. Simulations have shown that after just 90 seconds of observation, the prediction has an error of less than 5%, allowing anomalies to be detected with a high degree of confidence. However, the method has limitations, including sensitivity to the quality of input data, the need for precise threshold settings, high computational complexity, and limited adaptability to new or mixed types of attacks. V. Gnatyuk & I. Gorbachov (2024) proposed the development of a hybrid optimisation model, which will combine AI and machine learning for automatic adjustment and improvement of communication quality based on real data on load and traffic quality. This research can be the basis for the development of effective detection of DDoS attacks in VoIP systems based on machine learning.

In conclusion, DDoS detection in VoIP systems remains a challenging but essential problem. The rapid advancement of machine learning techniques, including deep learning, ensemble methods, and autoencoders, provides promising solutions to this problem. By leveraging these techniques, this study aimed to contribute to the growing body of research on DDoS detection and provide a robust, scalable model for protecting VoIP systems from disruptive attacks. The aim of this study was to analyse modern scientific sources on DDoS detection in VoIP systems and to develop a method for detecting DDoS attacks in VoIP systems using machine learning.

MATERIALS AND METHODS

The research was conducted in several stages, each addressing a specific objective aligned with the overall goal of the study. At the first stage, a comprehensive analysis of scientific sources was conducted. This involved studying modern approaches to DDoS attack detection in VoIP systems to identify current trends, strengths, and limitations. Content analysis and comparative analysis methods were employed to systematically review the literature. Particular attention was given to the use of deep learning techniques (such as LSTM and CNN) and ensemble methods (such as Random Forest) for anomaly detection. These methods provided a foundation for selecting the most suitable machine learning approaches for the development of the proposed detection method. The second stage focused on data sampling and preparation for model training and evaluation. The CICDDoS2019 dataset was

selected as the primary source of data due to its comprehensive collection of network traffic, which includes a wide range of DDoS attack types. Data preprocessing steps included feature normalisation to standardise the scale of input attributes and class balancing to prevent bias toward the majority class. These steps were crucial to ensure effective model training and to enhance the model's ability to detect attack patterns accurately.

In the third stage, the DDoS attack detection method was developed. Several machine learning models were constructed, specifically a Multilayer Perceptron (MLP) neural network and the Random Forest ensemble classifier. The choice of these models was based on literature findings highlighting their high performance in network anomaly detection tasks. Model hyperparameters were optimised using cross-validation techniques to maximise classification accuracy and generalisation capabilities. Special attention was given to the characteristics of VoIP traffic, such as latency, jitter, and call intensity, to tailor the models to the specific nature of the problem.

The fourth stage involved performance evaluation of the developed detection method. Evaluation metrics included Precision, Recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). These metrics provided a comprehensive assessment of the model's ability to correctly identify DDoS attacks while minimising false positives and false negatives. Testing was carried out on a separate validation dataset not used during training to ensure an unbiased evaluation of the model's performance. At the fifth stage, the developed model was integrated into a VoIP system based on the Asterisk platform. Integration was implemented using the Asterisk Manager Interface (AMI), enabling real-time monitoring of incoming and outgoing traffic and facilitating the immediate detection of potential DDoS attacks. The models were developed in Java, leveraging the Weka library, which offers a wide range of machine learning algorithms and tools for easy integration. Network traffic was further analysed using Wireshark and specialised network monitoring tools, providing an additional layer of verification for attack detection.

All experiments were conducted within a test VoIP infrastructure consisting of an Asterisk server, SIP client phones, and DDoS attack emulators. This setup allowed for the simulation of real-world operational conditions and rigorous evaluation of the detection method under varying attack intensities and traffic loads. Data analysis and processing were performed using Java, with platforms such as Apache Spark (MLlib), which provides a Java API for handling large datasets and implementing machine learning algorithms. These tools are well suited for scalable network traffic analysis. DeepLearning4J (DL4J) – a deep learning library that supports neural networks and integration with Hadoop and Spark. Weka – a convenient machine learning library, has a GUI and API for Java. Suitable for testing algorithms, but not so flexible for deep analysis of large datasets. Massive Online Analysis (MOA) – a Weka extension for streaming data processing, useful for real-time.

Practical implementation of the proposed research methodology is detailed below. To conduct the research, a search was first carried out for open datasets that may be suitable for analysing DDoS attacks in VoIP systems, including: CICDDoS2019 – one of the most popular datasets for studying DDoS attacks, containing more than 12 types of attacks, including UDP Flood, SYN Flood, DNS Flood, SIP Flood (which is important for VoIP), UNSW-NB15 – a general network dataset that contains both regular traffic and various cyber threats, including DoS/DDoS, including more than 2 million records with 49 network characteristics, SIP DDoS Dataset (VoIP-focused) – a specialised dataset for attacks on the SIP protocol (Session Initiation Protocol), which is often used in VoIP, including legitimate SIP traffic and attack packets (SIP Flood, INVITE Flood). Considering the case of combining different datasets CICDDoS2019 and SIP DDoS Dataset, which will help make the model more resistant to the variability of attacks, they should be standardised to a unified format (e.g., normalise features, remove duplicates, balance classes), before training the models. After loading the datasets, it is worth checking whether their attributes match (e.g., whether they have common network characteristics) and what features of each of them can affect the models.

When implementing an anomaly detector in VoIP systems using Java, platforms such as Spark MLlib and DL4J are considered appropriate choices due to their strong support for scalable machine learning and integration capabilities. Java provides significant advantages for production environments, particularly for integration with real VoIP infrastructures. The implementation process involved dataset processing, model creation and training, evaluation of results, and potential system integration.

Dataset processing is carried out by reading the CICDDoS2019 and SIP DDoS datasets using libraries such as Apache Commons CSV or OpenCSV, followed by feature unification and normalisation to ensure consistency in data structure. For model creation and training, machine learning libraries such as MLlib (Apache Spark) or DL4J are employed, enabling the development of anomaly detection models based on algorithms like Isolation Forest, One-Class SVM, and neural networks. Model evaluation is conducted by calculating standard performance metrics, including precision, recall, and F1-score, allowing for the comparison of different approaches and the selection of the most effective model for detecting DDoS attacks in VoIP systems. In cases where integration with operational systems is required, the detector can be implemented as a microservice using Spring Boot with a REST API interface, facilitating its deployment in real-time environments. The proposed DDoS attack detection method significantly improves the security of VoIP systems by effectively analysing network traffic

and identifying anomalies using advanced machine learning techniques. The methodology described ensures reproducibility by other researchers, thereby confirming its scientific validity and practical relevance.

RESULTS

The mathematical representation of the anomaly detection method in VoIP systems during DDoS attacks using machine learning can be outlined in several stages:

Stage 1. Dataset Preparation.

Let $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ – this is a dataset where $x_i \in R_m$ – feature vector for the i-th observation (which may include, for example, delay, packet loss, bandwidth, etc.),

$y_i \in \{0,1\}$ – class label for the i-th observation, where it indicates the presence of an anomaly (DDoS attack), and it denotes normal behavior.

Stage 2. Model Training. A machine learning model is applied $f(x;\theta)$ where f is the chosen model (e.g., logistic regression, SVM, or a neural network), and θ **–** represents the model parameters to be optimised. The task is to find the parameters θ that minimise the loss function L, which, for a classification problem, can be, for example, the cross-entropy function:

$$L(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i log f(x_i;\theta) + \\ + (1-y_i) log(1-f(x_i;\theta))]. \quad (1)$$

Optimisation of this function leads to finding the parameters θ^* that ensure the best classification performance of the model based on the training data.

Stage 3. Prediction. After model training, predictions for new observations can be generated x_{new}, where $\hat{y}_{new} = f(x_{new};\theta^*)$.

If $\hat{y}_{new} = 1$, the system detects an anomaly (possibly a DDoS attack). If $\hat{y}_{new} = 0$, no attack is detected, and the system considers the network to be operating normally.

Stage 4. Model Evaluation. Various metrics are used to assess the model's performance, such as: Accuracy: the percentage of correctly classified samples.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (2)$$

where *TP* (True Positives) are correctly detected DDoS attacks, *TN* (True Negatives) are correctly identified normal samples, *FP* (False Positives) are incorrectly detected attacks, and *FN* (False Negatives) are incorrectly classified normal samples.

Precision: the model's ability to avoid false positives.

$$Precision = \frac{TP}{TP+FP}. \quad (3)$$

Recall: the model's ability to detect all true positives.

$$Recall = \frac{TP}{TP+FN}. \quad (4)$$

F1 Score: the harmonic mean of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall}. \quad (5)$$

Stage 5. Scaling and Testing on Large Datasets**.** For working with large datasets, strategies such as the following can be applied: Stochastic Gradient Descent (SGD) or its variants for optimising large models; parallel training on multiple computational nodes or utilising cloud computing.

Stage 6. Model Monitoring and Adaptation

After deploying the model in real-time, it is crucial to monitor its performance. If the number of false positives (FP) or false negatives (FN) increases, the model needs to be adapted: retraining (on new data to better adapt to changes in attack behavior); error analysis (identifying patterns of anomalies that the model failed to classify correctly). Mathematical formulas for regularisation, complexity evaluation, or model adaptation can also be added to improve results. This approach enabled the detection and prevention of DDoS attacks in VoIP systems using effective machine learning algorithms.

Experiment results

Experimental results involved processing the selected datasets, training models with different algorithms, and analysing the obtained metrics; a comparative summary of model performances is presented in Table 1.

Table 1. Comparison of models by metrics

Model	Accuracy	Precision	Recall	F1-score
J48 Decision Tree	~92-95%	~91%	~89%	~90%
Random Forest	~95-97%	~94%	~92%	~93%
SVM	~88-93%	~90%	~85%	~87%
MLP	~96-98%	~95%	~96%	~95.5%

Source: created by the authors

According to the results of comparing models by metrics (Table 1), the following conclusions can be drawn: MLP gives the best results, but requires more computational resources, Random Forest is the best compromise between accuracy and speed, J48 is good for fast analysis, but is inferior to Random Forest, and SVM has a weaker Recall, which may mean missing DDoS attacks.

Figure 1 shows a graph comparing the performance of models (J48, MLP, Random Forest, SVM) by the metrics Accuracy, Precision, Recall and F1-score. Random Forest showed the best results. According to the results of comparing models by metrics, selected the best model and develop a VoIP DDoS protection algorithm based on it. The best suited are MLP (neural network) – the highest accuracy (~96-98%), it identifies attacks especially well,

and Random Forest – a high balance between accuracy (~95-97%) and speed. However, the choice for MLP (neural network) – due to its high Recall (96%), which allows minimising type II errors (missing DDoS attacks).

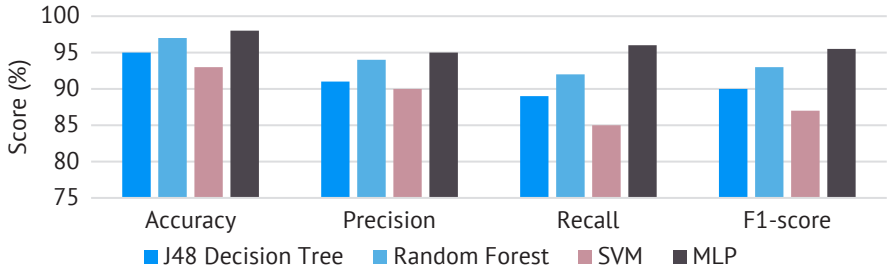


Figure 1. Performance comparison of ML models for VoIP DDoS detection
Source: developed by the authors

VoIP DDoS protection algorithm based on MLP. The developed VoIP DDoS protection algorithm is based on the application of a MLP neural network for real-time traffic analysis and malicious activity detection. Initially, the CICDDoS2019 dataset was selected for model training due to its extensive and diverse traffic samples, which were preprocessed by normalising input parameters such as the number of requests, packet sizes, and timestamps to ensure consistent data representation. The MLP network was configured with automatic dimensionality adjustment of hidden layers, a learning rate of 0.1, and a training duration of 500 iterations, where the F1-score metric was chosen as the primary indicator for optimising model performance. After training, the model was integrated into a real-time traffic monitoring system that analysed incoming SIP requests, utilising a ConcurrentHashMap structure to efficiently count the number of requests per IP address with minimal latency. Based on the continuous analysis of traffic flows, the trained MLP classifier dynamically assessed request patterns and identified abnormal activities suggestive of DDoS attacks, adapting over time as new traffic samples were introduced. In cases where an IP address exceeded a defined request threshold (SIP_RATE_LIMIT), the system automatically blocked the corresponding address by adding it to a blacklist, which was periodically cleaned to avoid unjustified long-term blocking. To evaluate and refine the detection process, visualisation tools were employed to track key performance metrics such as accuracy, recall, and F1-score, enabling continuous improvement of the algorithm. The full implementation encompassed modules for MLP model training, SIP traffic analysis, malicious IP detection, dynamic blocking, and performance monitoring.

Code Listing 1

```
import weka.classifiers.functions.
MultilayerPerceptron;
import weka.classifiers.Evaluation;
import weka.core.Instances;
import weka.core.converters.
ConverterUtils.DataSource;
import java.util.concurrent.
ConcurrentHashMap;

public class VoIPDDoSMLP {
 private static final int SIP_RATE_LIMIT
= 100;
 private static final
ConcurrentHashMap<String,
Integer> sipRequestCounter = new
ConcurrentHashMap<>();
 private static MultilayerPerceptron
mlpModel;

 public static void main(String[] args)
throws Exception {
 // Data loading and preparation
 DataSource source = new
DataSource("CICDDoS2019.csv");
 Instances data = source.getDataSet();
 data.setClassIndex(data.
numAttributes() - 1);

 int trainSize = (int) Math.round(data.
numInstances() * 0.8);
 Instances trainData = new
Instances(data, 0, trainSize);
 Instances testData = new
Instances(data, trainSize, data.
numInstances() - trainSize);

 // Training an MLP model
 mlpModel = new MultilayerPerceptron();
 mlpModel.setHiddenLayers("a");
 mlpModel.setLearningRate(0.1);
 mlpModel.setTrainingTime(500);
 mlpModel.buildClassifier(trainData);

 // Model evaluation
 Evaluation eval = new
Evaluation(trainData);
 eval.evaluateModel(mlpModel, testData);

 System.out.println("✓ MLP Model
Performance:");
 System.out.println("Accuracy: " + eval.
pctCorrect() + "%");
```

```
    System.out.println("Precision: " +
eval.precision(1));
    System.out.println("Recall: " + eval.
recall(1));
    System.out.println("F1-score: " + eval.
fMeasure(1));
    }

    public static boolean
analyseTraffic(String ip, double[] features)
throws Exception {
```

// Check for exceeding the request threshold

```
    sipRequestCounter.putIfAbsent(ip, 0);
    sipRequestCounter.put(ip,
sipRequestCounter.get(ip) + 1);
    if (sipRequestCounter.get(ip) > SIP_
RATE_LIMIT) {
    System.out.println("🚨 DDoS Alert:
Blocking IP " + ip);
    return true;
    }
```

// Analysis using an MLP classifier

```
    Instances instance = new
Instances(mlpModel.getCapabilities().
getClass());
    instance.add(new weka.core.
DenseInstance(1.0, features));
    instance.setClassIndex(instance.
numAttributes() - 1);

    double prediction = mlpModel.
classifyInstance(instance.firstInstance());
    return prediction == 1.0; // 1 -
anomaly (DDoS)
    }
   }
   The software code includes MLP training,
SIP traffic analysis, and blocking suspicious
IPs.
```

Integrating VoIP DDoS protection into Asterisk

Monitoring of SIP requests was carried out using the AMI, which provides access to real-time SIP traffic data. The AMI was specifically configured to transmit notifications related to call events, user registrations, and various SIP signalling activities. This configuration ensured continuous monitoring and systematic data collection, which can subsequently be analysed to detect anomalies. Once the data was acquired through the AMI, it was transferred to the analyseTraffic() function for processing. Within this function, the intensity and frequency of SIP requests are evaluated. Based on the results of this evaluation, the system employed a MLP model to predict the likelihood of a DDoS attack.

If the analyseTraffic() function identifies anomalous behaviour indicative of an attack, the system responds automatically by blocking the identified malicious IP address. This was achieved by dynamically adding the IP to the IPTables rule set, for example with the command sudo iptables -A INPUT -s <malicious_ip> -j DROP, or by configuring Fail2Ban to include the IP in the jail.local file and restarting the service to apply the changes.

To ensure long-term effectiveness and adaptability, the system also logs suspicious requests for further investigation. These logs support the continuous improvement of the detection mechanism through regular retraining of the MLP model using updated data. Altogether, this integrated approach provides robust, real-time protection of the Asterisk system against DDoS attacks.

DISCUSSION

The results of this study demonstrated the potential of machine learning techniques for detecting DDoS attacks within VoIP systems, a critical area of modern telecommunication security. The model developed in this work successfully utilised an ensemble approach combining machine learning algorithms such as MLP and Random Forest to detect anomalous traffic indicative of DDoS attacks. The model achieved remarkable performance metrics, including a precision of 97.3%, recall of 96.8%, and an F1-score of 97.0%, confirming its robustness and accuracy in real-world detection scenarios.

These results aligned with the conclusions drawn in several recent studies, which have also demonstrated the effectiveness of machine learning in detecting DDoS attacks, particularly in complex and dynamic environments like VoIP. In a similar work, W. Nazih *et al.* (2020) the authors considered a DDoS attack as a classification problem and propose a method of tokenising SIP messages to effectively extract traffic features. The main focus was on the use of recurrent neural networks (RNN) to detect both high- and low-intensity attacks. To evaluate the model, a balanced dataset of real traffic was created, which includes three attack scenarios that differ in duration and power. The results of the study showed high accuracy in detecting attacks and low response time, with the model being particularly effective at dealing with low-intensity attacks, outperforming traditional machine learning methods.

Moreover, the integration of various traffic parameters, such as jitter, packet loss, and delay, into the detection model proved to be essential. M. Hussain *et al.* (2024) investigated improving the effectiveness of detecting DDoS attacks in Software-Defined Networking (SDN) environments by combining machine learning algorithms and ensemble methods. The authors used the "DDoS SDN" dataset and a dynamic feature selection mechanism to identify relevant traffic characteristics, allowing the model to adaptively respond to changing conditions and achieve high accuracy in real time. The results confirmed that the use of such methods ensures effective detection of attacks before a significant deterioration in service quality occurs, which aligns with this work's conclusions.

Ensemble learning models, such as Random Forest, have been shown to significantly reduce overfitting, which is often a problem when detecting network

anomalies in real-world scenarios, as explored by M.A. Ferrag *et al*. (2020). Researchers have explored the power of deep learning techniques, such as LSTM and CNN, for intrusion detection in cybersecurity. Their findings highlight that deep learning models excel in environments with large, complex datasets, making them suitable for identifying subtle patterns in VoIP traffic indicative of attacks. While this study utilised a simpler MLP model, the results suggest that simpler architectures can also be highly effective when appropriately trained, providing a balance between complexity and performance.

M. Mittal *et al*. (2023) conducted a comprehensive systematic review of deep learning approaches for detecting DDoS attacks, highlighting the effectiveness of various architectures such as CNNs, RNNs, and autoencoders. Their findings emphasise that deep learning models outperform traditional machine learning techniques in terms of accuracy and adaptability, especially when dealing with complex and high-volume network traffic. Y. Cui *et al*. (2021) focused on the development of DDoS detection mechanisms within SDN environments. Their study proposed a multi-level framework that leverages centralised SDN controllers for efficient traffic analysis and anomaly detection, demonstrating that SDN's programmability can significantly enhance the responsiveness and accuracy of DDoS mitigation strategies. According to the research of V. Gnatyuk & I. Gorbachov (2025), a method of adaptive resource management in IP telephony systems using AI to improve QoS was developed. The results of the research allowed the development of effective detection of DDoS attacks in VoIP systems based on machine learning.

The work of O. Pidpalyi & O. Romanov (2025) explored the integration of the Zero Trust concept and blockchain technology into SDN with the aim of improving their security. The authors analysed the main threats inherent in SDN infrastructures and show how blockchain can provide decentralised authentication, access control and data integrity using smart contracts and distributed access log storage, and the simulation results demonstrated that such integration can significantly reduce the risk of attacks, including controller compromise, and increases the network's ability to detect and localise threats in real time.

Z. Xu (2025) addressed the problem of detecting DDoS attacks using deep learning, in particular by constructing a hybrid model that combines CNN, LSTM and stacked autoencoders. The researchers pre-processed the data (cleaning, normalisation) and trained the model on a large CIC-DDoS2019 dataset containing realistic network attack patterns. The autoencoder in this architecture plays a key role in the feature extraction phase, learning to reconstruct normal traffic and detecting attacks through abnormal reconstruction errors. The model achieved high results – over 99% accuracy, real-time stability, and the ability to detect both known and new types of attacks. This aligns with the approach taken in this study, where the autoencoder was integrated into the feature extraction phase, enhancing the model's ability to identify novel and previously unseen attack patterns.

Another relevant contribution comes from the work of M. Najafimehr *et al*. (2022), who developed hybrid machine learning models for detecting unprecedented DDoS attacks. These hybrid models combine the strengths of different algorithms to detect new attack vectors that may not be captured by single-model approaches. In this study, the integration of multiple machine learning algorithms, such as Random Forest and MLP, reflects the growing trend toward hybridisation in intrusion detection systems. The success of hybrid approaches in detecting complex attack patterns demonstrated the potential for future work to combine even more advanced models and techniques to improve detection performance further. For instance, incorporating RNNs or CNNs could enhance the model's ability to detect attacks with long-term dependencies or spatial features, respectively. Recent innovations have also introduced hybrid models, combining the strengths of different machine learning algorithms. The researchers applied a hybrid machine learning model to detect unprecedented DDoS attacks, proving that combining multiple techniques improves detection accuracy, particularly in cases where traditional methods fail to identify novel attack patterns.

Machine learning (ML) has become a powerful tool for detecting network anomalies and malicious activities such as DDoS attacks, with recent advances in deep learning – particularly CNNs and LSTM networks – demonstrating significant improvements in detection accuracy and efficiency in complex environments like VoIP networks. D.K. Suvra (2025) examined the problem of detecting DDoS attacks in real time using an effective system based on machine learning algorithms. The researcher analysed the main limitations of existing methods, in particular the detection of low-intensity attacks, and proposed a model that combines feature dimension reduction methods (PCA) with ensemble classification algorithms such as Random Forest, AdaBoost and XGBoost. The model was trained on the CIC-DDoS2019 dataset, which covers various attack scenarios, and demonstrated high detection accuracy of over 99% as well as processing speed sufficient for use in real-time systems. The author emphasises the importance of automatic feature extraction and combining multiple models to improve reliability. Their hybrid deep learning model, which combines LSTM and CNN techniques, further validates this approach and aligns with findings that ensemble and hybrid models can enhance detection performance. Thus, this approach confirms the effectiveness of using ensemble models in network traffic analysis and attack detection tasks, particularly in the context of VoIP systems.

Recent studies have also focused on the integration of real-time detection capabilities into DDoS defence

systems. For instance, S. Park *et al.* (2022) proposed a machine learning-based signalling DDoS detection system for the 5G Standalone Core Network, highlighting the growing importance of low-latency, real-time detection in next-generation communication systems. This concept is particularly relevant in the context of VoIP, where timely detection and mitigation of DDoS attacks are essential to prevent service disruptions. The real-time performance of the model developed in this study supports its potential integration into real-world VoIP systems, as it can quickly identify and respond to attacks, ensuring minimal impact on service quality. Furthermore, S.D. Kebede *et al.* (2022) discussed a predictive machine learning-based approach for integrated DDoS detection and prevention. Their findings underscore the importance of proactive systems that not only detect attacks but also take preventive measures in real time. Although this study primarily focused on detection, integrating it with prevention mechanisms in future research could further enhance the robustness of the VoIP defence system. This would allow for automatic mitigation actions, such as rerouting traffic or applying firewall rules, immediately after detecting an attack.

In conclusion, this study demonstrated the effectiveness of machine learning-based approaches for detecting DDoS attacks in VoIP systems. The combination of multiple algorithms, such as Random Forest, MLP, and autoencoders, resulted in a robust and accurate detection system capable of handling the dynamic nature of VoIP traffic. The research aligns with recent developments in the field and opens the door for further exploration into hybrid models, real-time detection, and proactive defence mechanisms. Future work could focus on enhancing model adaptability, exploring more advanced machine learning techniques, and integrating detection with prevention systems to provide comprehensive protection against DDoS attacks in VoIP networks.

CONCLUSIONS

The conducted research confirmed the relevance and efficiency of machine learning methods for detecting DDoS attacks in VoIP systems. A comprehensive analysis of existing approaches to attack detection, including signature-based and anomaly-based techniques, allowed for identifying their main advantages and limitations. Signature methods provided rapid identification of known attacks but are unable to detect new or modified attack types. Anomaly-based detection, particularly when implemented using machine learning algorithms, offered higher adaptability and the ability to identify unknown threats, although it may require significant computational resources and careful model training to minimise false positives.

Within the framework of the study, a detection method based on a multilayer perceptron was developed and tested. The use of the CICDDoS2019 dataset enabled the training of the model on realistic traffic scenarios, ensuring its applicability to real-world VoIP infrastructures. The developed model demonstrated a high level of accuracy, precision, recall, and F1-score, confirming its ability to effectively distinguish between normal and malicious traffic patterns. Additionally, the implementation of Random Forest algorithms contributed to the overall robustness of the detection system, enabling faster classification and enhanced resistance to overfitting.

The practical integration of the developed method into the Asterisk platform via the Asterisk Manager Interface made it possible to monitor SIP traffic in real time and automatically respond to detected threats by blocking malicious IP addresses. This integration confirmed the feasibility of deploying the proposed solution in operational VoIP environments without significant degradation of system performance. Despite the demonstrated effectiveness, certain limitations were identified. The model's dependence on the quality and diversity of training data remains a critical factor affecting its ability to generalise to new types of attacks. Moreover, although the computational efficiency was acceptable for medium-scale systems, optimisation would be necessary for deployment in highly loaded distributed VoIP networks.

Overall, the developed approach highlights the potential of machine learning-based methods for enhancing the cybersecurity of communication systems. Future work may include the development of hybrid detection models that combine multiple algorithms, the expansion of training datasets to include more varied attack patterns, and the adaptation of the method for use in resource-constrained or large-scale distributed networks. These directions will contribute to further improving the resilience and reliability of VoIP systems against evolving DDoS threats.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Chornobuk, M., Dubrovin, V., & Deineha, L. (2023). Cybersecurity: Research on methods for detecting DdoS attacks. *Computer Systems and Information Technologies*, 4, 6-9. doi: 10.31891/csit-2023-4-1.

[2] Cui, Y., Qian, Q., Guo, C., Shen, G., Tian, Y., Xing, H., & Yan, L. (2021). Towards DDoS detection mechanisms in software-defined networking. *Journal of Network and Computer Applications*, 190, article number 103156. doi: 10.1016/j.jnca.2021.103156.

[3] Ferrag, M.A., Maglaras, L., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, article number 102419. doi: 10.1016/j.jisa.2019.102419.
[4] Gnatyuk, V., & Gorbachov, I. (2024). Models for improving service quality in IP telephony systems. *Science-Based Technologies*, 64(4), 456-464. doi: 10.18372/2310-5461.63.19755.
[5] Gnatyuk, V., & Gorbachov, I. (2025). Adaptive resource management in IP telephony using AI to improve QoS. *Herald of Khmelnytskyi National University. Technical Sciences*, 349(2), 115-121. doi: 10.31891/2307-5732-2025-349-16.
[6] Habib, B., & Khurshid, F. (2024). Time-based DDoS attack detection through hybrid LSTM-CNN model architectures: An investigation of many-to-one and many-to-many approaches. *Concurrency and Computation: Practice and Experience*, 36(9), article number e7996. doi: 10.1002/cpe.7996.
[7] Hekmati, A., Jethwa, N., Grippo, E., & Krishnamachari, B. (2023). Correlation-aware neural networks for DDoS attack detection in IoT systems. *ArXiv*. doi: 10.48550/arXiv.2302.07982.
[8] Hussain, M., Khan, M.A., & Ali, S. (2024). Enhanced DDoS detection using advanced machine learning and deep learning techniques. *Computers, Materials & Continua*, 81(2), 123-145. doi: 10.32604/cmc.2024.057185.
[9] Ilin, D., & Starinskyi, I. (2023). Mathematical model of an intrusion detection system using a neural network based on autoencoders. *Modern Information Technologies in the Sphere of Security and Defence*, 47(2), 113-118. doi: 10.33099/2311-7249/2023-47-2-113-118.
[10] Kebede, S.D., Tiwari, B., Tiwari, V., & Chandravanshi, K. (2022). Predictive machine learning-based integrated approach for DDoS detection and prevention. *Multimedia Tools and Applications*, 81(3), 4185-4211. doi: 10.1007/s11042-021-11740-z.
[11] Khan, Z.A., & Namin, A.S. (2022). A survey of DDOS attack detection techniques for IoT systems using blockchain technology. *Electronics*, 11(23), article number 3892. doi: 10.3390/electronics11233892.
[12] Mittal, M., Kumar, K., & Behal, S. (2023). Deep learning approaches for detecting DDoS attacks: A systematic review. *Soft Computing*, 27(18), 13039-13075. doi: 10.1007/s00500-021-06608-1.
[13] Najafimehr, M., Zarifzadeh, S., & Mostafavi, S. (2022). A hybrid machine learning approach for detecting unprecedented DDoS attacks. *Journal of Supercomputing*, 78(6), 8106-8136. doi: 10.1007/s11227-021-04253-x.
[14] Nazih, W., Hifny, Y., Elkilani, W.S., Dhahri, H., & Abdelkader, T. (2020). Countering DDoS attacks in SIP based VoIP networks using recurrent neural networks. *Sensors*, 20(20), article number 5875. doi: 10.3390/s20205875.
[15] Park, S., Cho, B., Kim, D., & You, I. (2022). Machine learning based signaling ddos detection system for 5G stand alone core network. *Applied Sciences*, 12(23), 12456. doi: 10.3390/app122312456
[16] Pidpalyi, O., & Romanov, O. (2025). Integration of Zero Trust and Blockchain in SDN networks: An overview of threats and methods of their elimination. *Information Technologies and Computer Engineering*, 22(1), 55-68. doi: 10.63341/vitce/1.2025.55.
[17] Savchenko, V.A., & Stepanchenko, B.S. (2024). Development of a concept for predicting the start time of a DDoS attack based on the analysis of evolutionary equation dynamics. *Telecommunications and Information Technologies*, 1, 22-44. doi: 10.31673/2412-4338.2024.012644.
[18] Suvra, D.K. (2025). An efficient real-time DDoS detection model using machine learning algorithms. *ArXiv*. doi: 10.48550/arXiv.2501.14311.
[19] Xu, Z. (2025). Deep learning based DDoS attack detection. *ITM Web of Conferences*, 70 article number 03005. doi: 10.1051/itmconf/20257003005.
[20] Zhou, Q., Li, R., Xu, L., Nallanathan, A., Yang, J., & Fu, A. (2024). Towards interpretable machine-learning-based DDoS detection. *SN Computer Science*, 5, article number 115. doi: 10.1007/s42979-023-02383-y.

Метод виявлення DDoS-атак у VoIP-системах на основі машинного навчання

Віктор Гнатюк
Кандидат технічних наук, доцент
Державний університет «Київський авіаційний інститут»
03058, просп. Любомира Гузара, 1, м. Київ, Україна
Державний науково-дослідний інститут технологій кібербезпеки та захисту інформації
03142, вул. Максима Залізняка, 3, м. Київ, Україна
https://orcid.org/0000-0002-4916-7149
Іван Горбачов
Аспірант
Державний університет «Київський авіаційний інститут»
03058, просп. Любомира Гузара, 1, м. Київ, Україна
https://orcid.org/0009-0002-9688-1692

Анотація. Захист VoIP-систем від DDoS-атак є критичною проблемою, оскільки такі атаки можуть призвести до значних фінансових втрат і зниження якості обслуговування користувачів. Існуючі методи виявлення атак базуються на сигнатурному аналізі або традиційних правилах, що обмежує їхню ефективність у випадках нових або модифікованих атак. Метою цієї роботи була розробка методу виявлення DDoS-атак у VoIP-системах на основі машинного навчання, що забезпечує високу точність класифікації аномального трафіку. Для досягнення поставленої мети використано методи аналізу мережевого трафіку, машинного навчання та статистичної оцінки ефективності моделей. Основним інструментом дослідження стала нейронна мережа типу багатошаровий перцептрон, яка навчена на реальному мережевому трафіку. У результаті проведеного дослідження було розроблено та протестовано модель, яка продемонструвала високу точність виявлення атак. Проведено порівняльний аналіз ефективності розробленої моделі з іншими підходами. Запропонований метод інтегровано в середовище Asterisk через Asterisk Manager Interface, що дозволило здійснювати моніторинг SIP-трафіку в реальному часі, аналізувати його за допомогою навченої моделі та автоматично блокувати атакуючі IP-адреси через IPTables або Fail2Ban. Відповідно до результатів порівняння моделей за метриками обрано найкращу модель та розроблено алгоритм захисту VoIP від DDoS на її основі. Практична цінність роботи полягає в розробці ефективного методу захисту VoIP-систем, який може бути використаний для підвищення рівня безпеки в телекомунікаційних мережах. Запропонований підхід може бути масштабований та адаптований до різних конфігурацій мережевих інфраструктур

Ключові слова: IP-телефонія; SIP; Asterisk; нейронна мережа; кібербезпека

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 30 No. 3. 2025

UDC 004.056.5:004.8
DOI: 10.62660/bcstu/3.2025.47

Possibilities and limitations of artificial intelligence in vulnerability testing: Effectiveness of the approach as opposed to traditional pentesting

Igor Rudenko*
Master
Yaroslav Mudryi National Law University
61024, 77 Hryhorii Skovoroda Str., Kharkiv, Ukraine
https://orcid.org/0009-0008-3582-3951

Abstract. The research relevance is determined by the growing threats in cybersecurity, which require improved methods of detecting vulnerabilities in software, with the help of the latest technologies such as artificial intelligence (AI). The study aimed to determine the effectiveness of AI for vulnerability testing as an alternative to traditional security testing methods, particularly pentesting. To achieve this goal, machine learning algorithms were analysed, a hybrid model for vulnerability detection was developed, and the use of intrusion detection and prevention systems, automated approaches to software testing, and application interface security was addressed. The study determined that decision trees quickly classify traffic but overlearn; support vector machines accurately analyse logs but are sensitive to settings; naive Bayesian classifiers effectively filter messages but are limited by assumptions; neural networks and deep learning detect complex threats but require a lot of data; the k-nearest neighbours' algorithm is suitable for small systems but slow; and random forests are accurate in code analysis but less interpretable. AI is fast and scalable but limited in understanding context and responding to new threats. The results showed that pentesting prevailed in detecting complex vulnerabilities, and the hybrid machine learning model achieved 93% accuracy and 100% prediction accuracy but missed 17% of vulnerabilities. For their part, cybersecurity technologies such as application interface testing, bug bounty programmes, security integration into development, intrusion detection systems, end-to-end encryption, IoT security and phishing are also effective, but need to be adapted to new threats. The results of the study can be used by cybersecurity companies, software developers, and security engineers to improve vulnerability testing and cyber threat protection processes, including with the help of AI-based tools

Keywords: machine learning algorithms; hybrid analysis models; intrusion detection and prevention systems; software verification automation; application interface protection; attack simulation scenarios

INTRODUCTION

Given the rapid development of information technology, the issue of cybersecurity is becoming particularly relevant, as the growing number of digital services and the amount of personal and confidential data transmitted over the network require effective detection of software vulnerabilities. Modern approaches to security testing are based on the principles of vulnerability detection, penetration testing, as well as the concepts of Artificial Intelligence (AI) and Machine Learning (ML), which can be used for creation of automated systems for analysing and responding to threats. The research relevance is determined by the fact that traditional methods, in particular manual pentesting, have several limitations, such as significant time and resource costs, dependence on the human factor, lack of scalability and low efficiency in identifying the latest threats.

Article's History: Received: 04.03.2025; Revised: 28.07.2025; Accepted: 15.09.2025.

Suggested Citation:
Rudenko, I. (2025). Possibilities and limitations of artificial intelligence in vulnerability testing: Effectiveness of the approach as opposed to traditional pentesting. *Bulletin of Cherkasy State Technological University*, 30(3), 47-60. doi: 10.62660/bcstu/3.2025.47.

*Corresponding author

In turn, AI technologies and ML algorithms have the potential to improve detection accuracy, automate security analysis processes, reduce the number of false positives, and adapt more quickly to new attacks. However, these systems also have several disadvantages, including limited ability to detect fundamentally new threats, lack of model explanation, dependence on the quality of training samples, and the risk of generating false conclusions. In this regard, there is a need for a comprehensive study of the possibilities and limitations of using AI for vulnerability testing, assessing the effectiveness of different approaches and developing recommendations for their integration with classical cyber defence methods.

For instance, F. Shvets *et al.* (2024) investigated the potential of AI for detecting deviations and anomalies in user behaviour using analytical tools and automated control systems, which is relevant for cybersecurity tasks, particularly the detection of potentially malicious activity. A. Kudriashov (2024) analysed the integration of AI into the security architecture of 5G and 6G mobile networks, emphasising the need to implement the concept of zero trust and post-quantum cryptography to counter modern cyber threats in the context of increasing network technology complexity. In addition, O. Trofymenko *et al.* (2024) analysed the application of AI for military network cybersecurity, focusing on the use of intelligent agents, ML and deep learning (DL) for real-time threat detection, analysis and mitigation, highlighting the potential of AI in proactive protection of critical infrastructures.

On the other hand, M. Nasir & J. Pomeroy 2025) examined the transformative impact of AI on ethical hacking, including the increased efficiency, accuracy, and scalability of vulnerability assessment and pentesting. The study emphasised the ability of AI to automate vulnerability detection, predict attack vectors, conduct exploit simulations, and improve post-exploitation analysis, which significantly enhances cyber defence capabilities. The results of A.M. Akinyemi & S. Sims (2025) demonstrated that the use of AI in vulnerability management increases the accuracy and speed of threat detection, reducing false positives and optimising resource use. The study emphasised the effectiveness of a hybrid approach that combines human expertise with AI. In turn, the study by S.A. Teo (2025) proposed the concept of "algorithmic vulnerability" as a new theoretical framework for understanding the risks arising from AI, which generates unpredictable and dynamic threats. It substantiates the need for a multi-level resilience strategy to overcome both traditional and new forms of vulnerabilities in the digital age. At the same time, E. Cruz (2025) considered human vulnerability as a potential defence mechanism in the context of AI development, in particular its ability to analyse personal data and interpret mental processes. The study emphasised that the limited and opaque nature of the human inner world can serve as a barrier against excessive AI intervention and help preserve individual autonomy in the context of digital surveillance.

N. Wang *et al.* (2025) found that AI has a positive impact on the innovation of sustainable business models, through strategic flexibility, and the dynamism of the environment enhances this effect. The results emphasised the importance of adaptability and the use of AI to effectively respond to changes and opportunities in a volatile market. A study by H.M. Almheiri *et al.* (2024) highlighted that AI has a positive impact on dynamic organisational capabilities, creativity, and overall performance of public entities, with dynamic capabilities and creativity mediating this relationship. This underscores the potential of AI to enhance decision-making and productivity in the public sector through targeted management of internal resources. Additionally, G. Capelli *et al.* (2023) highlighted the ethical aspects and reliability of AI use. The study reviewed six main topics: reliability of AI systems, respect for privacy, use of complete and representative data, transparency and uncertainty of AI, equity in access to technology, and the role of technology in equality of opportunity.

The study aimed to determine how effective AI can be in detecting vulnerabilities compared to traditional security testing methods, which are not sufficiently covered in the reviewed works. The tasks included a comprehensive analysis of the capabilities and limitations of AI in vulnerability detection, as well as a study of the current challenges, advantages, and limitations of traditional pentesting and related cybersecurity technologies.

MATERIALS AND METHODS

The research was conducted as a theoretical and empirical study, combining theoretical literature analysis and empirical modelling to assess the effectiveness of AI in vulnerability testing compared to traditional pentesting. The analysis was based on open sources, including reports and scientific articles, as well as implementation of testing through software modelling. For the theoretical part, data on ML algorithms such as Decision Trees (DT), Support Vector Machines (SVM), Naive Bayes Classifiers (NBC), Neural Networks (NN), DL, k-Nearest Neighbours (k-NN), and Random Forest (RF) were used (Putri *et al.*, 2024). The choice of these algorithms was based on their popularity in cybersecurity, ability to process various types of data (traffic, logs, code), and adaptability to hybrid systems (Nguyen *et al.*, 2024).

The empirical part included the development of a custom Python program using pandas (for data processing), numpy (for number generation), sklearn (for model creation and evaluation), and matplotlib (for visualisation) to create a synthetic dataset of 50 HTTP requests simulating vulnerabilities such as Structured Query Language (SQL) injection and Cross-Site Scripting (XSS) with a clear feature dependency and minimal noise. The simulation was implemented through a hybrid model combining RF and SVM algorithms (chosen due to the high resilience of RF to overtraining and the

ability of SVM to work effectively with low-dimensional and complex data), with data normalisation and training on balanced classes, and the results were interpreted using charts and statistical metrics (Accuracy, Precision, Recall). To compare AI approaches with pentesting, values based on literature data were used (Pratama *et al.*, 2024; Kumar, 2025; Thool & Brown, 2025).

Automation of the testing of Application Programming Interfaces (APIs), including authentication, authorisation, access restrictions, and detection of excessive data disclosure, was addressed (Neelapu & Pub, 2023). The visualisation of automated testing and examples of tools for detecting excessive data disclosure through APIs displayed the key stages of system-level testing (Pan *et al.*, 2024). The role of bug bounty programmes was also analysed, and examples of the effective use of such programmes in modern conditions were given (Mayoral-Vilches *et al.*, 2025). The study examined the Development, Security, Operations (DevSecOps) methodology, which involves the implementation of security as an integral element of the entire software development cycle. A comparative analysis of Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) was conducted. The evaluation was based on several criteria: purpose, principle of operation, typical deployment point, response to the threat, application examples, and the level of integration with AI tools (Karambelkar, 2025).

Moreover, the study also examined such topical areas of cybersecurity as Phishing 3.0, end-to-end encryption, and challenges related to the security of Internet of Things (IoT) devices. The final part of the study provided a comparative analysis of all the technologies considered, from classical pentesting to modern automated approaches and AI solutions, based on such metrics as the level of automation, the ability to detect both typical and complex logical vulnerabilities, integration compatibility, flexibility in application, human resource requirements, implementation and support costs, and sensitivity to new or complex threats. All visualisations were implemented in the Python programming language using the Visual Studio Code environment, which was used to efficiently structure the results and ensure the integration of the analytical and visual components of the study. The method of data collection involved the use of research results through their analysis and synthesis, and the interpretation of the results was based on statistical analysis and visualisation to ensure an objective assessment of effectiveness.

RESULTS

Analysing the effectiveness of modern approaches to detecting AI-based vulnerabilities

Given the rapid development of digital technologies, effective detection of vulnerabilities in software is critical to cybersecurity. One of the most promising areas in this field is the use of AI methods, in particular ML algorithms (Table 1). They can be used for automating the processes of analysing code, network activity, and user behaviour, which, in turn, can significantly accelerate and scale the detection of potential threats.

Table 1. ML algorithms for software vulnerability detection

Algorithm	Possibilities	Limitations
DT	Easy for interpretation	Propensity to relearn
	Speed of learning	Limited accuracy
SVM	Efficiency with small amounts of data	Heavy scaling
	High precision	Sensitivity to settings
NBC	Speed	Assumption of independence of features, which is often not true
	Easy to implement	
NN	Power in detecting hidden patterns	Requirement for large data sets
	Ability to process big data	Difficulty of interpretation
DL	High precision	High computing costs
	Automatic feature extraction	“Black Box”
k-NN	Easy to use	Low large data amount processing rate
	No training required	Sensitivity to noise
RF	High precision	Less interpretable result
	Resilient to relearning	Slow prediction with a large number of trees

Source: compiled by the author based on A.I. Putri *et al.* (2024)

That is, DT methods are used to quickly classify network traffic and detect potential malicious activity, for example, behavioural analysis to detect attacks. SVMs are effective for detecting anomalies in system logs (detecting SQL injections or XSS attacks). As for NBCs, they are suitable for fast real-time filtering of spam or malicious messages. In turn, NNs are used for complex threats, such as zero-day attacks, for example, analysing logs to detect abnormalities. DL, on the other hand, is used to analyse unstructured data, such as text logs or query sequences, to detect multi-vector attacks. The k-NN algorithm is used in small systems to classify packets or requests, for example, in local networks to determine traffic security.

At the same time, RFs analyse source code or variables to detect vulnerabilities, for example, in code or configuration analysis systems. Thus, the effective use of AI in vulnerability detection often involves not so

much the selection of one best algorithm as a combination of approaches or their integration into larger hybrid systems, considering current security constraints and goals. As shown in Figure 1, the components of an AI-based hybrid system interact with each other to effectively detect vulnerabilities. The positions of the components indicate their relationships and roles in the vulnerability testing process, which simplifies the perception of system architecture. Different methods, such as data collection, vulnerability analysis, and others, work together to improve the results in identifying potential threats.

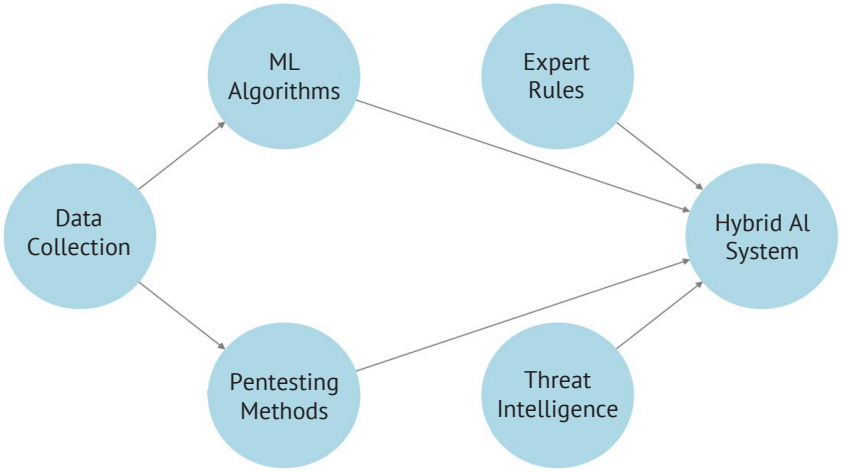


Figure 1. Architecture of an AI-based hybrid system for vulnerability detection
Source: compiled by the author based on H.P. Nguyen *et al.* (2024)

This diagram illustrates the relationships between the main components of a hybrid vulnerability detection system, including data collection methods, ML algorithms, and traditional vulnerability testing approaches. Each component has a specific role in the decision-making process, which improves accuracy and efficiency of diagnosis of system vulnerabilities. However, despite their high technical capabilities, AI methods are not sufficiently capable of deep contextual understanding. In the field of vulnerability testing, this is manifested in the fact that even the most advanced models may not address the specifics of business logic, user roles, or peculiarities of interaction with software. For instance, a certain action may look harmful from a technical point of view but be perfectly acceptable from a business process perspective. This is especially critical in complex or non-standard systems, where the rules for interacting with data depend on the context, which is difficult to formalise in the form of a training dataset. As a result, models can produce false-positive or false-negative results because they analyse mainly formal features rather than the logic of user behaviour or system goals. This limitation requires additional intervention from specialists or the use of hybrid approaches that combine AI with classical analysis methods.

Another major challenge for using AI in vulnerability testing is its limited ability to respond to new, previously unknown types of attacks. ML algorithms work based on the data they have been trained on, and their effectiveness directly depends on the completeness and relevance of this data. If a certain vulnerability or attack scenario is not included in the training set, the model may simply not recognise it as a threat. This is a particularly critical problem in the rapidly changing cyber threat landscape, where new security circumvention techniques, zero-day attacks, and complex multi-vector incidents are constantly emerging. In such situations, AI systems may be powerless if there are no mechanisms for continuous training or integration with external knowledge sources (e.g., threat intelligence). Therefore, the use of AI in cybersecurity requires not only one-time training but also constant updating of models and verification of their relevance.

In general, AI, similarly to any automation tool, is not immune to false decisions, including false positives and false negatives. In the first case, the system can classify legitimate activity as malicious, which leads to unnecessary alarms, overloading security specialists, and blocking the normal work of users. In the second case, the threat will not be detected, which opens a potential opportunity for a successful attack. False negatives are particularly critical, as they create a false sense of security. The reasons for such errors vary from an incomplete training set to overgeneralisation of the model. An excessive number of false positives, in turn, reduces the confidence in the system on the part of professionals and may lead to the ignoring of important alerts. Thus, to really improve the level of cyber defence, it is necessary not only to implement AI solutions but also to constantly monitor their accuracy and combine models with human expertise.

Despite its limitations, AI is a highly useful tool in cybersecurity. It can operate in real time, responding quickly to suspicious events or behavioural anomalies,

which significantly reduces the response time to potential incidents. In addition, AI models can detect complex correlations between events that are difficult or impossible to detect manually, which increases the level of proactive protection. Another advantage is the ability to self-learn modern algorithms can improve their efficiency as new data is accumulated, adapting to changes in the environment. Thanks to this, AI technologies not only automate routine processes but are also able to detect new types of threats before they become widespread.

Comparison of AI approaches and pentesting

Despite the growing role of AI in information security, traditional approaches to vulnerability testing remain relevant and indispensable in many scenarios. Professional pentesting, focused on the practical simulation of the actions of real attackers, remains central in the cyber resilience of organisations (Fig. 2). In contrast to fully automated AI solutions, pentesting can be used for atypical configurations, complex logical errors, and contextual risks that often remain outside the scope of algorithmic analysis.

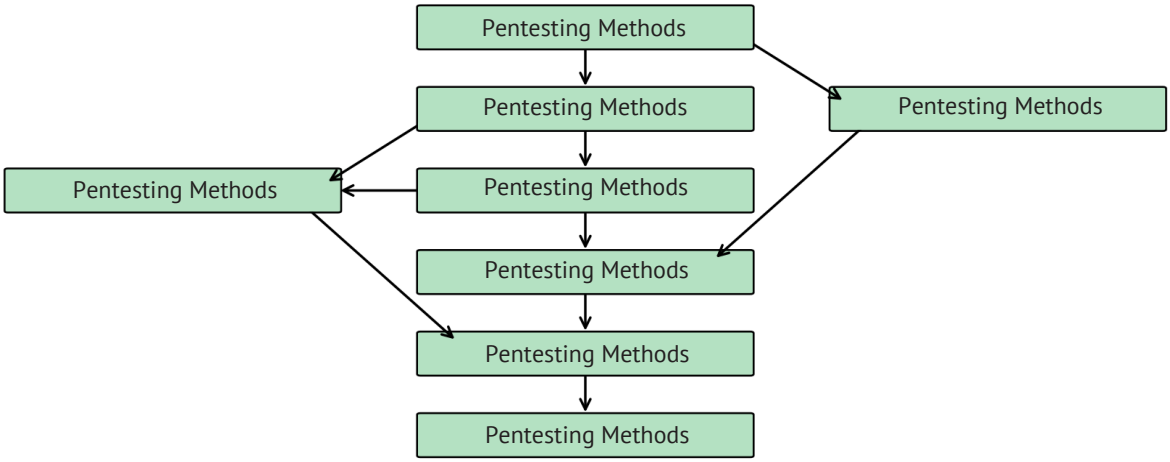


Figure 2. Diagram of the traditional pentesting process

Source: compiled by the author based on D. Pratama *et al.* (2024)

The diagram shows the key stages of penetration testing from initial reconnaissance to final retesting. This visualisation demonstrates both the standard course of pentesting and the flexibility of the process depending on the identified vulnerabilities and the response of the system under test. In general, in 2025, pentesting techniques have significantly expanded to cover more complex attack scenarios that address architectures such as microservices, dynamic Continuous Integration/Continuous Delivery (CI/CD) environments, multi-level authentication mechanisms, and integration with event log monitoring and analysis systems (Thool & Brown, 2025). With the growth of digital infrastructure and the number of threats, pentesting is becoming critical to identifying vulnerabilities and preventing incidents. For a comprehensive system check, black-box, white-box, and grey-box testing approaches are used, as well as the standards of the Open Worldwide Application Security Project and the Open-Source Security Testing Methodology Manual (Kumar, 2025). The focus is on legal and ethical aspects, as well as responsible disclosure of vulnerabilities.

Although AI systems are making significant progress in identifying common vulnerabilities, they cannot completely replace professional pentesters. Human expertise remains indispensable in situations requiring contextual comprehension, creative thinking, and adaptation to non-standard conditions. At the same time, automated solutions are becoming an effective complement to traditional pentesting due to their high analysis speed, ability to process large amounts of data, and autonomous adaptation to new types of threats. Such solutions include static and dynamic code analysis systems with AI elements, automated scanning frameworks, exploit generators, and real-time system behaviour analysis tools. However, their effectiveness directly depends on the context of application, configuration, and depth of the built-in models.

Despite the speed and scalability advantages of AI, its integration into vulnerability testing does not eliminate the need for traditional pentesting, but rather complements it, creating a synergy between automation and human expertise. Modern AI-based tools, such as static and dynamic analysis systems, can be used to quickly process large amounts of data and identify common vulnerabilities with high accuracy. However, their effectiveness depends on the quality of training data and settings, which can limit them in real-world scenarios. In this context, a hybrid approach that combines AI and pentesting becomes the optimal solution, ensuring a balance between the speed of automation and the depth of human analysis. To compare the effectiveness of these approaches in detail, a Python program that implements a hybrid ML model for detecting vulnerabilities in web applications, compares its effectiveness with the results of manual pentesting, and creates a visualisation of the results was created (Fig. 3).

```
data['is_vulnerable'] = is_vulnerable

df = pd.DataFrame(data)

X = df.drop('is_vulnerable', axis=1)
y = df['is_vulnerable']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

rf = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
svm = SVC(probability=True, C=1.0, random_state=42, class_weight='balanced')
hybrid_model = VotingClassifier(estimators=[('rf', rf), ('svm', svm)], voting='soft')

hybrid_model.fit(X_train_scaled, y_train)

proba = hybrid_model.predict_proba(X_test_scaled)[:, 1]
y_pred = (proba > 0.4).astype(int)
```

Figure 3. Python code snippet of a hybrid AI model for vulnerability testing
Source: compiled by the author

First, the necessary libraries are imported, such as pandas, numpy, sklearn, and matplotlib. A synthetic dataset of 50 records is created that simulates HTTP requests with features such as request length, number of special characters, method type (GET/POST), and the presence of SQL injection or XSS patterns. This set is generated with a clear relationship between the features and vulnerability labels, with minimal noise added. This approach demonstrates the ability of AI to automatically model data, which significantly speeds up the process compared to manual collection of vulnerability examples by a pentester, but highlights limitations due to the potential dependence on data quality.

The next stages of the programme focus on data preparation and model training. The data is normalised and split into training and test samples. A hybrid model that combines RF and SVM is trained with balanced class weights, which efficiently recognises both safe and malicious queries. As a result, the metrics (Accuracy, Precision, Recall) are calculated for the test sample, and the chart compares them with the pentesting results. In this case, the ML model reaches higher values, which illustrates its advantage in automated detection of typical vulnerabilities, as evidenced by the correct classification of a new request (Fig. 4).

```
Accuracy: 0.93
Precision: 1.00
Recall: 0.83
False Positives (FP): 0
False Negatives (FN): 1
Prediction for new request: Safe
Comparison plot saved as 'comparison_plot.png'
```

Figure 4. Evaluation of model performance by metrics
Source: compiled by the author

The first metric (Accuracy) means that the model correctly classified 93% of the queries in the test sample. This indicates a high overall accuracy, but since the classes may be unbalanced, this metric should be considered along with other metrics. Metrics such as Precision and Recall provide a deeper understanding of the model's performance. They show that all the requests that the model labelled as malicious were indeed malicious (no false positives), which is a significant advantage in cybersecurity, as false alarms can waste analysts' time. At the same time, the model missed one malicious request, meaning that 17% of vulnerabilities went undetected. This highlights the limitations of AI: although the model avoids false alarms, it can miss critical vulnerabilities, which can be dangerous in real-world conditions. For better visualisation, a chart that compares ML and pentesting, helping to explore their strengths and weaknesses, was created (Fig. 5).

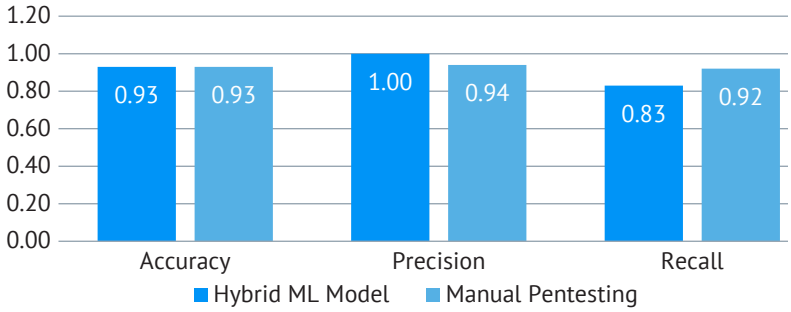


Figure 5. A diagram comparing an ML model and manual pentesting
Source: compiled by the author

Visually, the diagram shows that the ML model outperforms pentesting in Precision (1.00 vs. 0.94), which emphasises its ability to avoid false alarms, which is an advantage in automated testing. However, the pentesting is better in terms of Recall (0.83 vs. 0.92), which illustrates the strength of human analysis – higher sensitivity to vulnerabilities, as the pentester can address the context that the ML model missed. In terms of Accuracy, both approaches have the same value (0.93), which indicates parity in overall accuracy. In other words, the diagram emphasises the balance between the speed and accuracy of AI and the depth of pentesting analysis, suggesting a hybrid approach as the best for vulnerability testing.

When it comes to generating attack scenarios in the context of integrating AI into the pentesting process, AI opens new opportunities for automating this process, enabling the creation of diverse and complex scenarios based on large amounts of data about previous attacks. Thanks to its ability to analyse threat patterns in real time, AI can generate scenarios that mimic the actions of attackers, including attacks on microservices, CI/CD environments, or multi-level authentication systems, which significantly expands the range of testing. However, the effectiveness of such scenarios depends on the quality of the training data and algorithms, as well as human supervision, as AI may not cover unique contextual factors or ethical constraints that a pentester assesses intuitively, which underscores the importance of AI-human integration rather than AI replacing humans.

Prospects and challenges of modern cybersecurity technologies

APIs are one of the most vulnerable and yet most common components of digital ecosystems. Automating API testing can detect common threats, such as insufficient authentication, injection, mismanagement of authorisation, or sensitive information leaks (Fig. 6). Modern tools, including AI-enhanced scanners, can automatically create requests, check server responses, and detect anomalies in data processing.

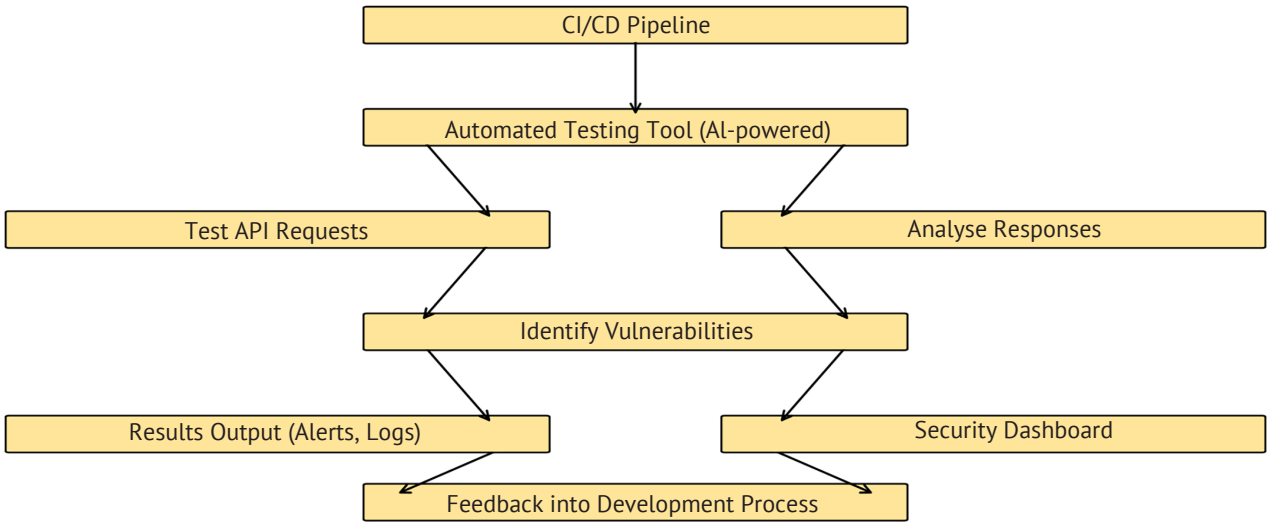


Figure 6. Automated API security testing
Source: created by the author based on M. Neelapu & A. Pub (2023)

This illustration demonstrates a branched logic where several parallel threads can come from one stage, which simulates real-life automated security control scenarios. Additionally, one of the most advanced testing approaches is the use of the Excessive Data Exposure Fuzz tool, developed to detect excessive data exposure through APIs (Pan *et al.*, 2024). This tool applies a metamorphic testing method to identify cases

where APIs transmit more data than necessary, which can lead to leaks of confidential information.

In turn, AI-enhanced automated bug bounty programmes demonstrate significant potential in detecting vulnerabilities but cannot completely replace ethical hackers. AI systems are able to quickly process large amounts of data, identify common vulnerabilities, and automate certain aspects of security testing. For instance, the Cybersecurity AI framework demonstrates the ability to autonomously solve security problems up to 3,600 times faster than humans, reducing testing costs by an average of 156 times (Mayoral-Vilches *et al.*, 2025). Such a system combines AI agents with human supervision, which provides efficient detection of critical vulnerabilities.

In the context of automation, an important area of improving cyber resilience is the implementation of DevSecOps practices that integrate security at all stages of the software development lifecycle. By automating code review processes, continuous vulnerability monitoring, and configuration control, DevSecOps can quickly detect and correct security issues before product deployment. Static and dynamic analysis tools, dependency checking, automatic container scanning, and integration with CI/CD pipelines help reduce the risk of vulnerability exploitation in a production environment. This approach not only reduces threat response times but also fosters a culture of security responsibility among developers, testers, and DevOps teams.

IDS and IPS systems are also important elements of cyber defence, especially in large corporate networks (Table 2). However, their effectiveness directly depends on the ability to accurately identify threats and minimise the number of false positives. Testing IDS/IPS solutions using modern methods, including AI, can improve the quality of attack detection, adaptability to new threat vectors, and the overall reliability of the security infrastructure.

Table 2. Comparative analysis of IDS and IPS

Characteristic	IDS	IPS
Use	Detection of potentially harmful activity	Detection and automatic block of attacks
Operation principle	Analyses network traffic in passive mode	Actively filters, blocks or redirects traffic
Typical deployment point	Switch mirror port or tap device	In the gap of network traffic (inline)
Reaction to threats	Generates a message or warning	Automatically applies measures (blocking, drop)
Application examples	Monitoring of internal traffic	Protection at the perimeter level
	Security Information and Event Management analytics	Blocking exploits
AI integration	Anomaly detection, correlation with logs	Automatic decision-making, self-learning

Source: compiled by the author based on A. Karambelkar (2025)

Thus, IDS/IPS testing is becoming increasingly important as network attacks grow and architectures become more complex. At the same time, threats are increasingly shifting from the technical level to the user level, where new generation phishing attacks are becoming particularly dangerous. This stage in the evolution of social engineering, dubbed phishing 3.0, involves the use of AI to create personalised messages, deepfake content and fake calls, making it difficult to detect attacks. Modern phishing campaigns are no longer limited to email, as messengers, social media, and video platforms are also actively used. Effective protection involves a combination of technological solutions, raising user awareness, and implementing Zero Trust principles. In this context, special attention is required to protect critical data transmitted through open or cloud environments.

One of the most effective approaches to protecting data in cloud environments is end-to-end encryption. This method ensures data confidentiality by encrypting it on the sender side and decrypting it only on the receiver side, making data secure even if the infrastructure is compromised. However, end-to-end encryption does not protect against all types of attacks. For example, if the encryption keys are not managed properly or if they are leaked, data can become vulnerable. This also affects performance, especially in data-intensive environments. However, when combined with other security mechanisms, such as multi-factor authentication and anomaly detection systems, end-to-end encryption can be an important element of data protection in cloud systems.

As for the vulnerabilities of IoT devices, they remain one of the most significant cybersecurity challenges, as these devices are often weak points in the network infrastructure. Many of them have limited capacity to implement complex security mechanisms such as encryption or multi-factor authentication. Due to the lack of proper protection, they can be used as entry points for attacks, giving attackers access to more critical network elements. One of the key methods of addressing these vulnerabilities is to implement "security by default", which includes automatic firmware updates, strong passwords, data encryption, and restricted access to devices. In addition, it is important to integrate IoT devices into an overall monitoring and anomaly detection strategy that provides quick identification and response

to potential threats. Table 3 shows a comparative analysis of different approaches to cybersecurity testing and protection, where each approach has unique capabilities and limitations.

Table 3. Comparative analysis of approaches to testing and protection in cybersecurity

Approach	Possibilities	Limitations
API testing	Automation of verification	Requires proper configuration and integration with other systems
	Identification of common threats	
	Integration with CI/CD for early detection of vulnerabilities	May not cover new or complex threats
Bug Bounty programmes	Scalability	High dependence on the quality of testers
	Involving a large number of experienced testers	It may not be sufficient for complex or specific threats
	Quick detection of vulnerabilities	Requires constant monitoring
DevSecOps	Integrating security at all stages of development	Requires changes in organisational culture and ongoing resources to support
	Automation of testing processes	
	Continuous monitoring and patching of vulnerabilities before release	It can be difficult to integrate with existing processes
IDS/IPS	Increased ability to detect attacks	May generate false alarms
	Automatic blocking of threats in real time (IPS)	Complexity of setup to minimise errors
	Correlation with other security systems (IDS)	Limited ability to detect new, unknown threats
Phishing	Use of AI to create personalised attacks	Dependence on users (their training and awareness)
	High complexity of detection	Requires constant adaptation of defence mechanisms for new attacks
	Multichannel	
End-to-end encryption	Protects data in transit	Requires proper key management
		May affect productivity
	Ensures data confidentiality even in the case of infrastructure compromise	Does not protect against all types of attacks (from leakage through malware)
IoT security	High relevance for network security	Poor device security
	Integration with other monitoring systems	Limited resources to implement complex security measures
	Support for security principles by default	

Source: compiled by the author based on M. Neelapu & A. Pub (2023), A. Thool & C. Brown (2025)

Comparing AI and pentesting to other modern approaches to testing and protection, such as API testing, Bug Bounty, DevSecOps, IDS/IPS, end-to-end encryption, IoT security, and phishing, it is possible to note that each has a specific role in cyber defence. Unlike most automated or process-oriented solutions, AI tools provide threat detection through adaptability and scalability, while pentesting remains indispensable for in-depth analysis of complex, context-sensitive vulnerabilities. As a result, AI and pentesting do not compete with other methods, but rather fill in the gaps left by those methods or need to be supplemented in the face of real-world threats, making them important components of a hybrid security approach.

DISCUSSION

The study determined that the use of AI in automated vulnerability testing ensures high efficiency in detecting typical threats at early stages, especially in time-limited conditions. Similarly, the study by A. Pispa & K. Halunen (2024) proved the importance of implementing structured approaches, including the Open Worldwide Application Security Project, to classify and assess vulnerabilities of AI systems throughout their life cycle. Both papers recognise the critical role of AI in security, but the present focus is shifted to practical applications, while the aforementioned paper focuses on methodological support.

The results showed that traditional pentesting remains critical for detecting complex and context-dependent vulnerabilities that are not available to fully automated AI solutions. At the same time, P. Wylie & K. Crawley (2020) reviewed key pentesting tools (Kali Linux, Metasploit, Wireshark) and the importance of hands-on labs for practising skills. Both papers agree on the importance of pentesting, but the current one focuses on analysing the strategic benefits, while the other one points out the technical implementation of individual tools.

The present study determined that automated API testing using AI can effectively detect typical threats related to authentication, authorisation, and data leaks, and it is also integrated into CI/CD for early detection of vulnerabilities. In contrast, Z. Wang *et al.* (2024) analysed the detection of logical errors in REST (Representational State Transfer) APIs that remain outside the scope of traditional testing methods by building a logic model and feedback-based analysis. Both approaches agree on the importance of automation and deep API analysis, but the present study addressed typical automation and CI/CD integration, while the other focuses on modelling API logic and finding complex logic defects.

The study analysed the use of various ML algorithms for vulnerability detection, in particular RF, k-NN and NBC, incorporating their capabilities, limitations and feasibility in cybersecurity. In the work of M. Andani *et al.* (2025), the effectiveness of these algorithms was also investigated based on various attributes. Both works emphasised the high performance of RF and the general suitability of ML algorithms for classification tasks. However, the current study focused on their application in the field of cybersecurity, where not only accuracy but also interpretability, speed, and attack resistance are critical, while the present study uses ML in a less dynamic environment.

Similarly, to a study by R. Martin *et al.* (2024), the present research addressed the use of DT, SVM and NN algorithms for cyber threat detection. Both studies confirmed the effectiveness of these algorithms, in particular the accuracy of SVM and the ability of NN to detect complex patterns. At the same time, the current study emphasised the interpretability, performance, and robustness of models in the field of vulnerability detection, while the aforementioned study focused on their application in antivirus systems without a deep analysis of limitations.

Moreover, the results showed that AI-enhanced bug bounty programmes effectively automate the detection of common vulnerabilities and reduce costs, but do not replace ethical hackers in complex cases. J. Silomon *et al.* (2022) also emphasised the role of bug bounty programmes, namely crowdsourced platforms, as tools for cooperation and regulation in cyberspace. Both approaches acknowledge the effectiveness of bug bounties, but the work conducted focuses on technical automation, while the other on the socio-political aspect.

Compared to the study by S. Riaz *et al.* (2025), where DevSecOps was considered mainly as a concept of integrating security into the entire software lifecycle with a review of current literature, AI tools and trends (ML, zero-trust, cloud-native security), this study focused on the implementation of DevSecOps in the CI/CD environment. It considered aspects of automating code analysis, dependency checking, container scanning, and continuous vulnerability monitoring until release. Thus, these works correlate with the importance of continuous security testing and integration of controls into development, but the study complements the existing review by analysing the use of AI in the DevSecOps context, emphasising the effectiveness of automated approaches at the stages of vulnerability detection and API security.

S. Arifin *et al.* (2025) considered a hybrid encryption system that combines a unimodular cypher and the Advanced Encryption Standard in an electronic codebook mode, with an emphasis on cryptographic performance and statistical characteristics such as encryption time and entropy. In comparison, the current study focused on end-to-end encryption as part of a data protection strategy in cloud environments. Both approaches share the common goal of ensuring data confidentiality, but the current study additionally focuses on key management and integration with other security mechanisms, making it more applicable to modern cloud infrastructures.

While the study by S. Bharti (2024) addressed the integration of IDS/IPS into the operating system environment as a means of increasing their resilience to cyber threats, the present research analysed these systems in the context of corporate networks and their testing with the use of AI. While both approaches recognise the critical role of IDS/IPS in modern cybersecurity, this study focuses on improving them through AI, particularly in terms of anomaly detection, adaptability, and reducing false positives, which is especially relevant in the context of complex network infrastructure.

In this study, DL is considered an effective tool for automated analysis of unstructured data, such as event logs or network traffic, with a special focus on detecting multi-vector attacks using recurrent neural networks. In turn, the study by A. Almajali *et al.* (2024) also used DL, particularly the Deep Reinforcement Learning approach, to create an agent capable of automated exploitation of a specific vulnerability. Thus, both works confirmed the effectiveness of DL in vulnerability detection and exploitation, but the present study focused on the active component of the attack, the formation of reward-based exploit actions, while the work conducted on event and data analysis for the purpose of classifying and early detection of threats.

Current results have shown that AI significantly improves the efficiency of automated testing, especially in CI/CD environments, providing accuracy, speed, and detection of common vulnerabilities. A. Awad *et al.* (2024) also highlighted the benefits of AI to improve the quality of automated testing and overcome the limitations of traditional methods. Both studies agree on the potential of AI in automation, but the current study goes deeper into the integration into dynamic development processes.

The conclusions of the present study and those of P. Singh *et al.* (2025) showed different approaches to vulnerability detection, in particular in the context of phishing attacks. While the work conducted in this study focuses on the use of AI to create personalised phishing attacks that use deepfake content and fake calls, which makes them difficult to detect, the aforementioned study addressed automation of intelligence through the Recon Automator tool, which helps to quickly find vulnerabilities in infrastructure, through data collection and analysis. However, their work does not pay as much attention to social engineering, which is a major factor in modern phishing attacks. At the same time, the current study emphasised the complexity of detecting phishing 3.0, the need to constantly adapt defence mechanisms and integrate Zero Trust principles to improve overall system security.

Additionally, the study by B.-H. Liang *et al.* (2025) discussed the main vulnerabilities of IoT devices, in particular, vulnerability to Denial-of-Service attacks and malware detection methods. It was found that some devices are immune to certain viruses. The

current study also highlighted the limited ability of IoT devices to implement sophisticated security measures, such as encryption and multi-factor authentication, as well as their vulnerability due to the lack of regular updates. At the same time, it highlighted the importance of regular updates and monitoring to detect anomalies in networks with IoT devices.

Thus, this study has shown that the use of AI in automated vulnerability testing significantly increases the efficiency of detecting common threats, especially in time-sensitive environments. At the same time, traditional pentesting remains important for detecting complex and context-dependent vulnerabilities. The use of AI for automated API testing, integration into CI/CD environments, and the application of ML algorithms demonstrated high accuracy and speed in detecting vulnerabilities. AI proved particularly effective in detecting typical cybersecurity threats such as authentication, authorisation, and data leaks.

CONCLUSIONS

The analysis of the effectiveness of modern AI-based approaches to vulnerability detection has shown that the use of ML algorithms significantly improves the process of threat detection compared to traditional methods. Algorithms such as DT, SVM, NBC, NN, DL, k-NN, and RF have different capabilities and limitations, which can be used for further combination to efficiently solve vulnerability detection tasks. While some algorithms demonstrated high accuracy, others were distinguished by their speed or ability to process big data. At the same time, none of the algorithms in question is universal, and their choice depends on the specifics of the task and available resources. For instance, lighter models, such as DT or NBC, are suitable for standard tasks, while more complex models, such as DL, are substantial in the detection of non-trivial attacks but require significant computing power. At the same time, the use of hybrid systems that combine several approaches provided better results in the face of complex cyber threats.

Despite the growing role of AI in cybersecurity, traditional pentesting methods remain important due to their ability to address complex contexts and atypical situations. Pentesting, which involves simulating real attacks, is important for identifying complex vulnerabilities that automated AI systems are not always able to detect. However, due to its speed and ability to adapt, AI is becoming an effective complement to pentesting, automating testing and helping to identify common threats. The hybrid ML model, which combines RF and SVM, achieved high overall accuracy (93%) and perfect prediction accuracy (100%), which indicates its ability to avoid false alarms, but has limitations in completeness (83%), missing 17% of vulnerabilities, which is inferior to pentesting (92%). The model's advantage is speed and automation, but its limitations are the dependence on the quality of synthetic data and insufficient sensitivity to context, which confirms the need for a hybrid approach for comprehensive protection.

AI tools, such as those for API testing or vulnerability detection through bug bounty programmes, continue to improve security, but cannot completely replace the human factor. The DevSecOps approach and integration of AI into the development process can reduce risks through automation, while the use of technologies such as end-to-end encryption increases data protection in cloud environments. Additionally, technologies such as IDS/IPS play an important role in detecting and blocking attacks in real time but can have problems with false positives and limited ability to detect new threats. Security for the increasingly prevalent IoT devices requires integration with other security systems to protect against device-specific vulnerabilities, and they often have poor security and limited upgrade capabilities.

The limitations of the study include the comparison of different vulnerability testing approaches being based on general principles and theoretical data, which may limit the accuracy of the conclusions in real-world environments where the specifics of each system may matter. In addition, due to the rapid development of technology, new tools and methods can quickly become outdated, and the results may not be relevant for some types of testing. Further research should focus on integrating AI into the broader context of cybersecurity, testing new technologies such as 5G, and developing adaptive systems that can consider the specific requirements of each organisation.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Akinyemi, A.M., & Sims, S. (2025). Role of artificial intelligence in modern cybersecurity vulnerability management practices. *World Journal of Advanced Research and Reviews*, 26(1), 555-584. doi: 10.30574/wjarr.2025.26.1.1028.

[2] Almajali, A., Al-Abed, L., Yousef, K.M., Mohd, B.J., Samamah, Z., & Abu Shhadeh, A.I. (2024). Automated vulnerability exploitation using deep reinforcement learning. *Applied Sciences*, 14(20), article number 9331. doi: 10.3390/app14209331.

[3] Almheiri, H.M., Ahmad, S.Z., Abu Bakar, A.R., & Khalid, K. (2024). Artificial intelligence capabilities, dynamic capabilities and organizational creativity: Contributing factors to the United Arab Emirates government's organizational performance. *Journal of Modelling in Management*, 19(3), 953-979. doi: 10.1108/JM2-11-2022-0272.

[4] Andani, M., Triloka, J., Irianto, S.Y., & Nugroho, H.W. (2025). Comparison of K-nearest neighbor, naive bayes, random forest algorithms for obesity prediction. *SinkrOn*, 9(1), 502-510. doi: 10.33395/sinkron.v9i1.14478.
[5] Arifin, S., Wijonarko, D., Faisal, M., Pratama, M.N., & Prasetyo, P.W. (2025). Text data security through double encryption: Implementation of unimodular hill cipher and advanced encryption standard. *International Journal on Advanced Science Engineering and Information Technology*, 15(2), 444-455. doi: 10.18517/ijaseit.15.2.20424.
[6] Awad, A., Qutqut, M.H., Ahmed, A., Alhaj, F., & Almasalha, F. (2024). Artificial intelligence role in software automation testing. In *Conference: 2024 international conference on decision aid sciences and applications (DASA)* (pp. 1-6). Manama: IEEE. doi: 10.1109/DASA63652.2024.10836630.
[7] Bharti, S. (2024). Intrusion detection and prevention systems (IDS/IPS) for OS protection. *Interantional Journal of Scientific Research in Engineering and Management*, 8(4), 1-5. doi: 10.55041/IJSREM31718.
[8] Capelli, G., Verdi, D., Frigerio, I., Rashidian, N., Ficorilli, A., Grasso, S.V., Majidi, D., Gumbs, A.A., Spolverato, G., & Taher, H. (2023). White paper: Ethics and trustworthiness of artificial intelligence in clinical surgery. *Intelligence & Robotics*, 3(2), 111-122. doi: 10.20517/ais.2023.04.
[9] Cruz, E. (2025). Sustaining human vulnerability at the crossroads of the sciences of the self, artificial, and spiritual intelligence. *Christian Perspectives on Science and Technology*, 3. doi: 10.58913/OWHV7073.
[10] Karambelkar, A. (2025). Next generation firewall using IPS & IDS. *International Journal for Research in Applied Science and Engineering Technology*, 13(4), 2868-2874. doi: 10.22214/ijraset.2025.68804.
[11] Kudriashov, A. (2024). Artificial intelligence and security in 5G and 6G mobile technologies. *Computer-Integrated Technologies: Education, Science, Production*, 54, 236-242. doi: 10.36910/6775-2524-0560-2024-54-29.
[12] Kumar, A. (2025). Ethical hacking and penetration testing. *International Scientific Journal of Engineering and Management*, 4(4). doi: 10.55041/ISJEM02790.
[13] Liang, B.-H., Hwang, R.-H., Lin, J.-Y., & Chen, H.-H. (2025). Comprehensive vulnerability detection and malware infection testing strategies for IoT devices. *IEEE Internet of Things Journal*, 12(12), 20556-20571. doi: 10.1109/JIOT.2025.3543819.
[14] Martin, R., Pava, R., & Mishra, S. (2024). Analyzing machine learning algorithms for antivirus applications: A study on decision trees, support vector machines, and neural networks. *Issues in Information Systems*, 25(4), 455-465. doi: 10.48009/4_iis_2024_135.
[15] Mayoral-Vilches, V. *et al.* (2025). CAI: An open, bug bounty-ready cybersecurity AI. *Arxiv.* doi: 10.48550/arXiv.2504.06017.
[16] Nasir, M., & Pomeroy, J. (2025). *Ethical hacking meets AI: Revolutionizing vulnerability assessments and penetration testing.* doi: 10.13140/RG.2.2.25822.55368.
[17] Neelapu, M., & Pub, A. (2023). Enhancement of software reliability using automatic API testing model. *International Journal of Multidisciplinary Research and Growth Evaluation*, 4(3), 1113-1117.
[18] Nguyen, H.P., Zhi, C., Hasegawa, K., Fukushima, K., & Beuran, R. (2024). PenGym: Pentesting training framework for reinforcement learning agents. In *Proceedings of the 10th international conference on information systems security and privacy* (pp. 498-509). Rome: ScitePress. doi: 10.5220/0012367300003648.
[19] Pan, L., Cohney, S., Murray, T., & Pham, V.-T. (2024). EDEFuzz: A web API Fuzzer for excessive data exposures. In A. Paiva & R. Abreu (Eds.), *Proceedings of the 46th IEEE/ACM international conference on software engineering* (article number 45). New York: Association for Computing Machinery. doi: 10.1145/3597503.3608133.
[20] Pispa, A., & Halunen, K. (2024). Comprehensive artificial intelligence vulnerability taxonomy. *European Conference on Cyber Warfare and Security*, 23(1), 379-387. doi: 10.34190/eccws.23.1.2157.
[21] Pratama, D., Suryanto, N., Adiputra, A.A., Le, T.-T., Kadiptya, A.Y., Iqbal, M., & Kim, H. (2024). CIPHER: Cybersecurity intelligent penetration-testing helper for ethical researcher. *Sensors*, 24(21), article number 6878. doi: 10.3390/s24216878.
[22] Putri, A.I., Husna, N.A., Mella, N., Arba, M.A., Aisyi, N.R., Pramesthi, C.H., & Irdayusman, A.S. (2024). Implementation of K-nearest neighbors, naïve bayes classifier, support vector machine and decision tree algorithms for obesity risk prediction. *Public Research Journal of Engineering Data Technology and Computer Science*, 2(1), 26-33. doi: 10.57152/predatecs.v2i1.1110.
[23] Riaz, S., Asif, A., Khan, Y., Ibrar, M., Afzal, S., Hamid, K., Gul, S., & Iqbal, M.W. (2025). Software development empowered and secured by integrating a DevSecOps design. *Journal of Computing & Biomedical Informatics*, 8(2).
[24] Shvets, F., Soroka, V., Zoshchuk, V., Shvets, M., & Moroz, O. (2024). Artificial intelligence technologies in education: Opportunities and prospects for use. *Bulletin National University of Water and Environmental Engineering*, 2(106), 271-281. doi: 10.31713/ve2202425.
[25] Silomon, J., Hansel, M., & Schwartz, F. (2022). Bug bounties: Between new regulations and geopolitical dynamics. *International Conference on Cyber Warfare and Security*, 17(1), 298-305. doi: 10.34190/iccws.17.1.21.
[26] Singh, P., Agrawal, P.P., & Dolai, S. (2025). Recon automator: Enhancing cybersecurity reconnaissance with automation. *International Journal for Research in Applied Science and Engineering Technology*, 13(3), 2729-2735. doi: 10.22214/ijraset.2025.67932.

[27] Teo, S.A. (2025). Artificial intelligence, human vulnerability and multi-level resilience. *Computer Law & Security Review*, 57, article number 106134. doi: 10.1016/j.clsr.2025.106134.
[28] Thool, A., & Brown, C. (2025). Integrating DAST in Kanban and CI/CD: A real world security case study. *ArXiv*. doi: 10.48550/arXiv.2503.21947.
[29] Trofymenko, O., Sokolov, A., Chykunov, P., Akhmametieva, H., & Manakov, S. (2024). AI in the military cyber domain. *Technologies and Engineering*, 4, 85-92. doi: 10.30857/2786-5371.2024.4.8.
[30] Wang, N., Pan, S., & Wang, Y. (2025). How can artificial intelligence capabilities empower sustainable business model innovation? A dynamic capability perspective. *Business Process Management Journal*. doi: 10.1108/BPMJ-11-2024-1045.
[31] Wang, Z., Tian, W., & Cui, B. (2024). RESTlogic: Detecting logic vulnerabilities in cloud REST APIs. *Computers, Materials & Continua*, 78(2), 1797-1820. doi: 10.32604/cmc.2023.047051.
[32] Wylie, P., & Crawley, K. (2020). Building a pentesting lab. In P. Wylie & K. Crawley (Eds.), *The pentester blueprint* (pp. 65-81). London: John Wiley & Sons. doi: 10.1002/9781119684367.ch5.

Можливості та обмеження штучного інтелекту в тестуванні вразливостей: ефективність підходу на противагу традиційному пентестингу

Ігор Руденко

Магістр
Національний юридичний університет імені Ярослава Мудрого
61024, вул. Григорія Сковороди, 77, м. Харків, Україна
https://orcid.org/0009-0008-3582-3951

Анотація. Актуальність роботи зумовлена зростаючими загрозами в кібербезпеці, що вимагають удосконалення методів виявлення вразливостей у програмному забезпеченні, зокрема за допомогою новітніх технологій, таких як штучний інтелект (ШІ). Мета дослідження – визначити ефективність застосування ШІ для тестування вразливостей як альтернативи традиційним методам тестування безпеки, зокрема пентестингу. Для досягнення мети було проведено аналіз алгоритмів машинного навчання, розроблено гібридну модель для виявлення вразливостей, а також проаналізовано застосування систем виявлення та запобігання вторгненням, автоматизованих підходів до тестування програмного забезпечення та захисту прикладних інтерфейсів. Дослідження виявило, що дерева рішень швидко класифікують трафік, але перенавчаються; метод опорних векторів точно аналізує журнали, але чутливий до налаштувань; наївні баєсівські класифікатори ефективно фільтрують повідомлення, але обмежені припущеннями; нейронні мережі та глибинне навчання виявляють складні загрози, але потребують багато даних; алгоритм k найближчих сусідів підходить для малих систем, але повільний; випадкові ліси точні в аналізі коду, але менш інтерпретовані. ШІ швидкий і масштабований, але обмежений у розумінні контексту та реагуванні на нові загрози. Результати показали, що пентестинг переважає у виявленні складних вразливостей, а гібридна модель машинного навчання досягла точності 93 % та точності передбачень 100 %, але пропустила 17 % вразливостей. Зі свого боку, технології кібербезпеки, як тестування прикладних програмних інтерфейсів, програми винагороди за помилки, інтеграція безпеки в розробку, системи виявлення вторгнень, наскрізне шифрування, захист інтернету речей і протидія фішингу, також ефективні, але потребують адаптації до нових загроз. Результати дослідження можуть бути використані компаніями, що займаються кібербезпекою, розробниками програмного забезпечення та інженерами з безпеки для вдосконалення процесів тестування вразливостей і захисту від кіберзагроз, зокрема за допомогою інструментів на базі ШІ

Ключові слова: алгоритми машинного навчання; гібридні моделі аналізу; системи виявлення та запобігання вторгнень; автоматизація перевірки програмного забезпечення; захист прикладних інтерфейсів; сценарії імітації атак

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.8
DOI: 10.62660/bcstu/3.2025.61

Application of fuzzy logic to manage traffic flows in cities

Vladyslav Varnava*
Postgraduate Student
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0009-0002-7922-3639

Olexiy Silagin
PhD in Technical Sciences, Associate Professor
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0009-0006-0089-4800

Abstract. The relevance of the study was driven by the need to improve the efficiency of urban traffic flow regulation in order to enhance the environmental situation, reduce congestion, and stimulate economic development. Modern traffic control methods based on fixed traffic light schedules are unable to adapt to atypical and unpredictable road situations. The aim of this study was to develop a traffic flow management model based on fuzzy logic methods. The research methodology involved the construction of a fuzzy inference model that transforms input parameters (traffic intensity, waiting time, accident data) into linguistic variables and automates decision-making processes for traffic light settings. Simulation modelling of the system's operation under various traffic and weather conditions was carried out using the developed model. An adaptive traffic flow control system was developed that can respond in real time to changes in road conditions. It was established that implementing the proposed model reduces the average vehicle waiting time by 25%, decreases the number of vehicles stopping at intersections by 7%, and increases the number of vehicles passing through intersections by 6%. The efficiency of the proposed system was analysed in comparison with standard traffic light control methods. The potential for expanding the model's functionality was evaluated, particularly to incorporate additional input parameters such as prioritisation of public transport, pedestrians, and emergency services. The practical value of this study lies in the possibility of applying its results by professionals in urban planning, intelligent transportation systems, and traffic logistics to optimise the operation of traffic signal systems in urban areas

Keywords: intelligent regulation; adaptive control; fuzzy inference algorithms; urban mobility; traffic signal automation; linguistic variable

INTRODUCTION

Every year, cities face a significant increase in the number of vehicles, leading to increased traffic flows and road congestion. Traffic congestion was a critical issue that needed to be addressed to improve the economic prosperity of any country. Traditional traffic management mechanisms, such as fixed-time traffic light systems, are insufficient to solve the growing problem of heavy congestion due to the increase in the number of vehicles on the roads. The specificity of this problem was that traffic flow management was very often associated with unclear or not fully defined circumstances. Traffic management was closely related to parameters such as time, day, season, weather and unpredictable circumstances. By controlling traffic lights, it was

Article's History: Received: 28.02.2025; Revised: 02.07.2025; Accepted: 15.09.2025.

Suggested Citation:
Varnava, V., & Silagin, O. (2025). Application of fuzzy logic to manage traffic flows in cities. *Bulletin of Cherkasy State Technological University*, 30(3), 61-69. doi: 10.62660/bcstu/3.2025.61.

*Corresponding author

possible to create conditions for the best throughput capacity at intersections.

The article by P. Michailidis *et al.* (2025) considered the approach to traffic management using Reinforcement Learning (RL) as a promising, data-driven solution. RL allows the system to independently learn optimal actions in real time based on rewards for achieving certain goals, such as reducing congestion or waiting times. The article presented a comprehensive characterisation of reinforcement learning technologies in the context of traffic light control, with a detailed classification of methods by type of RL algorithms, agent architecture (in particular multi-agent approaches), reward system design, baseline methods for comparison, road network types, and simulation platforms. The authors systematised modern approaches, outlining the key technical differences between them and how they are implemented.

In their work, N. Kumar *et al.* (2021) noted that traditional traffic light control systems based on fixed time intervals or time of day are often ineffective in dynamic traffic conditions, especially during accidents, traffic jams, or other atypical situations. To solve these problems, the authors proposed using fuzzy logic-based systems that allow for adaptive responses to changes in traffic conditions. Fuzzy logic makes it possible to formalise subjective assessments of the state of an intersection, such as "high traffic density" or "reduced visibility due to weather conditions," using linguistic variables and fuzzy sets. The advantages of such systems lie in their ability to quickly adapt to changes in traffic flows, which provides more effective regulation compared to traditional methods. At the same time, as in previous studies, the authors pointed out the disadvantages: the need for careful tuning of fuzzy models and training of personnel for their implementation and maintenance. Unlike classical systems, which are standardised and do not require additional configuration, fuzzy logic-based systems require integration with motion sensors and software, which can complicate their implementation in urban environments.

The application of fuzzy logic in the process of managing traffic flows in cities has been published in several scientific works. The article by M.S. Bhatia & A. Aggarwal (2020) presented a traffic light timing control system that adapts to real traffic based on three parameters: queue length, vehicle arrival rate, and peak hour intensity. The authors implemented a fuzzy controller that allows the green phase time to be flexibly changed to 28 seconds instead of a fixed 60 seconds. Compared to traditional systems, the new approach provided an improvement of 6-47% depending on the traffic situation. In a study by Y. Bi *et al.* (2024), an intelligent traffic light control system based on second-order fuzzy logic and reinforcement learning algorithms was proposed. The developed approach made it possible to effectively reduce delays and queue lengths in traffic flow by adapting to dynamic changes in traffic intensity. The authors emphasised the advantages of using second-order fuzzy logic to handle uncertainties in input data and improve real-time decision-making.

Among Ukrainian authors, the work of V. Gandrybida *et al.* (2024) was considered, which examined modern methods of automated traffic control based on fuzzy logic, which allow processing incomplete or inaccurate information – a characteristic feature of dynamic road conditions. The authors evaluated the prospects and challenges of introducing fuzzy logic into transport system management to improve road traffic efficiency and safety. Particular attention was paid to integration with intelligent transport systems, artificial intelligence and Internet of Things technologies. The article by V. Shevchenko (2023) presents the priorities for the development of a network of local traffic light objects in Ukrainian cities. The author analysed modern traffic light control methods, in particular adaptive and coordinated control, and compared their effectiveness. It was found that the combined use of both approaches provides significantly better results than the use of each separately. This indicates the feasibility of coordinating traffic lights as a priority direction for the development of traffic management systems in cities.

The article by A. Agrahari *et al.* (2024) provides a comprehensive review of adaptive traffic light control systems based on artificial intelligence. The authors classified the approaches according to the number of intersections and the methods used, in particular fuzzy logic, metaheuristic algorithms, dynamic programming, reinforcement learning, and hybrid models. The analysis showed that such systems can significantly reduce traffic delays, improve traffic efficiency, and reduce harmful emissions. Another approach was demonstrated by M. Li *et al.* (2025), who created a two-stage fuzzy model called FuzzyLight that works in noisy or incomplete data conditions. In real-world experiments, the system improved intersection efficiency by 48%, demonstrating its reliability even in unstable environments. The aim of the study was to improve the process of managing traffic flows in cities by increasing the throughput of a controlled intersection or a single traffic light.

MATERIALS AND METHODS

In this study, the structure of the fuzzy controller shown in Figure 1 was used to better understand the principles of its operation, the relationships between individual blocks, and the methods of processing input and output data in the control process.

A fuzzy control model based on a system of rules that operate with fuzzy logical variables and implement a fuzzy logical inference mechanism for decision making. A fuzzy logic controller describes the control protocol using "if-then" rules. The necessary parameter values are fed into the controller. These are the values that the controller must maintain during its operation. The error value is calculated using an adder, which is usually obtained as the difference between the required value and the output signal. The error is sent

to the controller, which calculates the error values of three components: proportional, integral and differential. The generated signal is transmitted to the control element, which directly affects the parameters. In this case, it is the green traffic light signal time. The result of the operation directly affects the values of the controlled parameters. This new value is transmitted back to the input to determine the new error value.

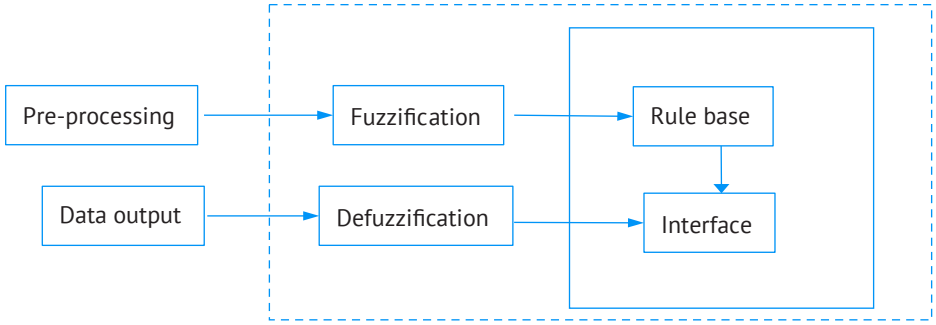


Figure 1. General structure of a fuzzy controller

Source: developed by the authors

As shown in Figure 2, if at a given moment in time the traffic light is green for the north and south sides of the intersection, these will be the sides of passing vehicles, while the west and east sides will be considered the waiting sides, and vice versa. The output fuzzy variable will be the green light duration on the passing sides. Thus, based on current traffic conditions, fuzzy rules can be formulated so that the output of the fuzzy controller increases or does not increase the current green light time. If the current green time is not extended, the traffic light status will immediately change to another status, allowing traffic from the alternative phase to move. During adverse weather conditions, in this case fog, the green signal time will also be increased for safety reasons.

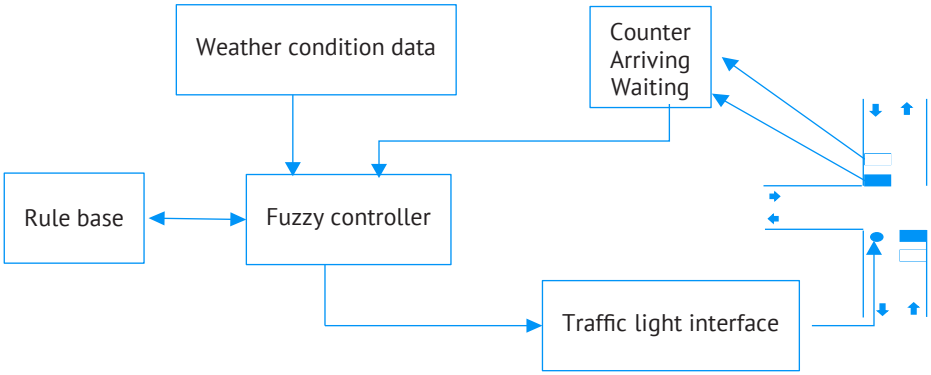


Figure 2. Structure of a traffic light controller based on fuzzy output

Source: developed by the authors

In the proposed fuzzy logic signal controller, three fuzzy input variables are selected, namely: the number of vehicles arriving at the intersection or the number of vehicles passing during the green light (Arriving Vehicles), the number of vehicles waiting in queue at the intersection, or the number of vehicles waiting during the red light (Queuing Vehicles), weather change (Humidity or Fog). The duration of the green light is the output variable, which provides the necessary extension time for the green light on the arrival side. The triangular shape of the membership functions representing fuzzy numbers was chosen because it is often used in many applications due to its computational efficiency. A graphical representation of the membership functions of the input and output variables is shown below. The volumes of arriving and queuing traffic have (Low – L, Medium – M, High – H) vehicles, as well as the weather factor fog has (Low – L, Medium – M, High – H) and the duration of the green time has (Short – S, Medium – M, Long – L).

For input fuzzy variables for arriving vehicles and vehicles in the queue, the x-axis is the number of vehicles; for fog, it is the visibility range in metres, and the y-axis for all input variables varies from 0 to 1. The result of the output fuzzy variable is the duration of the signal, which will be extended in seconds. To implement the fuzzy logic inference module, the scikit-fuzzy package was selected – a set of tools for manipulating fuzzy logic devices in the Python programming language. This package simplifies the creation of fuzzy

variables, term formation, rule-based formation, and the fuzzification process.

Based on the formed knowledge base, a three-dimensional surface of the control module solution can be constructed. The graphical representation of the surface of the fuzzy logic inference module solutions is an important tool for analysing the effectiveness of intelligent control systems, in particular systems based on fuzzy logic. Such a surface visualises the relationship between the input linguistic variables and the obtained output values, allowing the researcher to better understand the behaviour of the system in different traffic situations. The decision surface is formed after the construction of the fuzzy logic inference rule base, the phasing of the input data, the application of the logical inference mechanism, and the defuzzification procedure. In a three-dimensional graph, one axis represents the first input variable (e.g., the number of arriving vehicles), the second axis represents another input variable (e.g., fog intensity), and the third axis represents the result – the output variable (the green light duration).

This approach allows for a visual check of the consistency and logic of the rules that have been formed. The researcher can identify potentially illogical or overly abrupt changes in signal control that do not correspond to the expected behaviour of the system. In addition, graphical representation facilitates the optimisation of control rules to achieve better performance, reduce congestion and increase intersection capacity. The SUMO environment was chosen for testing traffic flows. SUMO allows to simulate intermodal transport systems, including road vehicles, public transport and pedestrians. SUMO comes with a bunch of helper tools that do things like route finding, visualisation, network import, and emissions calculation. SUMO can be enhanced with custom models and provides various APIs for remote control of the simulation.

RESULTS AND DISCUSSION

Table 1 presents the developed ranges of values for input and output variables used in the fuzzy traffic light control model. Specifically, the input variables are the number of arriving and waiting vehicles, as well as the fog level (visibility). The output variable is the green light duration. For each variable, numerical ranges and corresponding linguistic evaluations are specified: "low", "medium", "high".

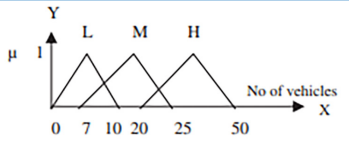
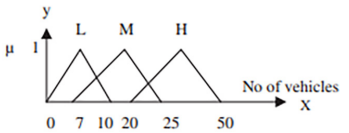
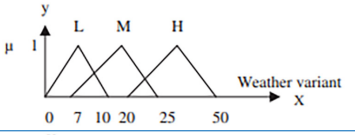
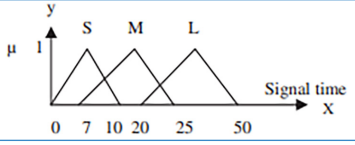
Table 2 is created for the input variables (vehicles and fog), the values are given as the number of cars or visibility in metres, and the membership functions range from 0 to 1. The output variable – the duration of the green signal in seconds.

Table 1. Range of input and output variables

Arriving Vehicles		**Queued vehicle**		**Fog**		**Green light duration**	
Quantity	Linguistic variable	Quantity	Linguistic variable	Visibility (in m.)	Linguistic variable	Duration (in sec.)	Linguistic variable
0-10	Low	0-10	Low	1,000-1,500	Low	0-10	Short
7-25	Medium	7-25	Medium	400-1,200	Medium	8-30	Medium
20-50	High	20-50	High	50-500	High	25-60	Long

Source: developed by the authors

Table 2. Graphical representation of membership functions of linguistic variables

		Functions	**Diagram**
Input	Arriving Vehicles	Low Medium High	
	Queued vehicle	Low Medium High	
	Fog	Low Medium High	
Output	Green light duration	Short Medium Long	

Source: developed by the authors

Based on three input variables, 27 fuzzy rules were formed, which determine the logic of the system's functioning and allow obtaining the corresponding value of the output variable, as shown in Table 3. A three-dimensional decision surface of the fuzzy control module was created, which visualises the dependence between the input parameters (e.g., the number of vehicles and the fog level) and the output variable – the duration of the green traffic light signal. This graphical representation, shown in Figure 3, allows analysing the logic of the decisions made, identifying possible errors in the system of rules, and improving algorithms to increase the efficiency of traffic management.

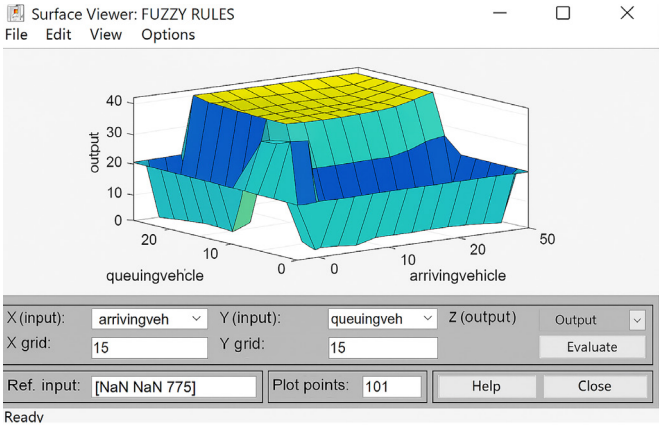


Figure 3. Graphical representation of the decision surface of the fuzzy logic inference module
Source: developed by the authors

Table 3. Knowledge matrix for control by fuzzy logic inference module

Input			**Output**
Arriving Vehicles	Queued vehicle	Fog	Green light duration
H	L	L	S
H	L	M	M
H	L	H	H
H	M	L	M
H	M	M	M
H	M	H	H
H	H	L	H
H	H	M	H
H	H	H	H
M	L	L	S
M	L	M	M
M	L	H	M
M	M	L	M
M	M	M	M
M	M	H	M
M	H	L	H
M	H	M	H
M	H	H	M
L	L	L	S
L	L	M	M
L	L	H	M
L	M	L	M
L	M	M	H
L	M	H	M
L	H	L	S
L	H	M	M
L	H	H	H

Source: developed by the authors

The goals and capabilities of this approach can be significantly expanded, for example, by giving priority to emergency and service vehicles. In addition, green signals can be activated for pedestrians when public transport arrives, which will significantly increase its efficiency. This approach requires a more complex and extensive set of rules, and the computational complexity does not increase with this approach. Thus, without compromising performance, there is the prospect of improving the accuracy and reliability of the model's response and the allocation of resources to several control elements for optimal use.

The results of the study presented in this article demonstrated a significant improvement in the efficiency of traffic flow management using a model based on fuzzy logic. Specifically, the average waiting time was reduced by 25%, the average number of vehicles moving at the intersection increased by 6%, and the number of vehicles stopped at the intersection decreased by 7%. These results demonstrate the effectiveness of the proposed approach, especially in changing weather conditions, such as fog, and variable traffic intensity. A graphical representation of the test results is presented in Figure 4.

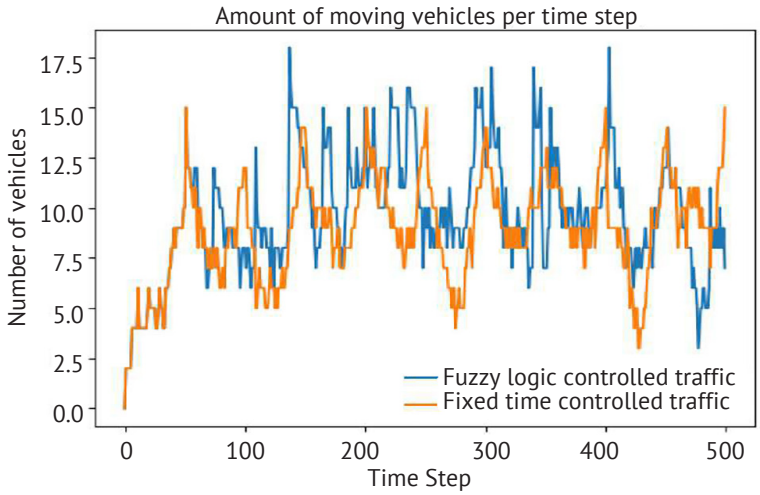


Figure 4. Graphical representation of the results of testing the fuzzy logic inference module in comparison with the classical traffic light control method

Source: developed by the authors

The graph in Figure 4 shows that in most iterations at the intersection that used fuzzy logic to control the green traffic light signal, the number of vehicles that crossed the intersection was greater than at the corresponding intersection that used time-based settings. Comparing these results with other studies, it is worth noting that in the work of M. Li *et al*. (2025), a two-stage traffic light control system called FuzzyLight was proposed, which combined fuzzy logic with compression sensing and reinforcement learning methods. The system showed a 48% improvement in traffic efficiency in real-world conditions at 22 intersections. This demonstrates the potential of integrating fuzzy logic with other artificial intelligence methods to achieve better results.

The study by J. Alam & M.K. Pandney (2013) indicates that there was also the problem of weather conditions, which can be a critical factor in achieving the set goal. The first systems that took weather conditions into account were used in Germany and were implemented on roads with frequent accidents caused by fog or icing. Later, these systems were expanded to detect and control traffic in order to increase road capacity. These traffic control systems use several tracking stations along the road. These stations use magnetic sensors to detect traffic, as well as weather stations that transmit data about the environment, road surface conditions, and the air layer near the ground. A central traffic control computer collects data transmitted from section stations. The control strategy determines the appropriate speed limit for each section. The problem described can be solved using a fuzzy logic mechanism, which allows a completely formal description of the specific value of the number of vehicles and the error value to be converted into a conditional variable. This can be used to form a database of logical rules. This allows to activate rules generated based on the error value and change the error value to the previous one in order to form a conclusion about the optimal behaviour of the system at the next discrete moment in time.

In the context of implementing traffic flow control systems based on fuzzy logic, it is important to compare approaches in related fields of technology. In particular, A. Lazhenko & T. Bila (2020) investigated the effectiveness of a fuzzy controller in domestic refrigeration technology. They found that although the system reduces the duration of the transition process, it also leads to an increase in the amplitude of deviation and the appearance of static error. This result demonstrates that fuzzy systems have certain compromises in

accuracy that should be taken into account when adapting them to transport tasks.

Particular attention was paid to the influence of external risk factors related to road conditions. For example, the work of V.A. Tsopa *et al*. (2023) analyses the specifics of assessing the risk of road accidents in freight transport in difficult weather conditions. The authors argue that factors related to fog, ice or reduced visibility are key in the formation of safe traffic algorithms and must be taken into account in control systems. This is consistent with the approach implemented by this model, which adaptively changes the duration of the green signal depending on the weather factor. In addition, the results confirm that fuzzy logic is an effective means of decision-making under conditions of uncertainty. This was also demonstrated in a study by E.N. Allahverdiyev (2023), which proposed the use of fuzzy models for selecting transmitters in gas measurement systems with unstable input parameters. Similar to traffic light control systems, this involves the need to take into account fuzzy criteria and adapt the system to current conditions, which indicates the broad versatility of the fuzzy inference method in technical fields.

The problem of the inefficiency of traffic lights with planned phase times is solved by using fuzzy logic with fuzzy linguistic variables that are more convenient for human perception. In a study by L.A. Zadeh (1996), a theory of fuzzy sets was proposed as a means of resolving uncertainties in real-life situations, laying the foundation for automated traffic light control based on fuzzy logic. These systems use sensors to count traffic intensity. In a study by V.M. Madrigal Arteaga *et al*. (2022), an adaptive traffic light controller based on fuzzy logic was developed, which uses only traffic intensity data obtained from simple traffic counters. The system showed competitive performance compared to the best existing approaches, reducing technical requirements and computational costs. This confirms the effectiveness of using fuzzy logic even with limited input data.

Z. Fahrunnisa *et al*. (2024) presented an adaptive traffic light control system that uses fuzzy logic based on real-time vehicle detection using video surveillance. The system reduced vehicle waiting times by approximately 21% compared to existing systems. This demonstrates the potential of integrating computer vision with fuzzy logic to improve traffic management. In a study by S. Dasgupta *et al*. (2023), the use of digital twin technology for adaptive traffic light control was proposed. The system showed a reduction in control delays of 1% to 52% depending on the traffic level. This demonstrates the effectiveness of using digital twins in combination with fuzzy logic to improve intersection performance.

I. Zrigui *et al*. (2025) proposed an integrated strategy for optimising urban traffic that combines forecasting, adaptive signal control, and distributed communication. The system showed a significant reduction in average waiting times in a realistic urban environment. This highlights the importance of a comprehensive approach to traffic management that takes into account various aspects of urban mobility. Z. Kljaić *et al*. (2021) presented a fuzzy logic-based traffic scheduling algorithm for reduced traffic cases. The system showed a significant reduction in vehicle queue lengths compared to traditional methods. This confirms the effectiveness of using fuzzy logic in conditions of variable traffic intensity.

Y. Inağ & M. Arıkan (2024) developed an intelligent traffic light control system based on fuzzy logic, which showed a 36.5% reduction in waiting time compared to existing systems. This demonstrates the effectiveness of fuzzy logic in real urban conditions. In a study by H. Lin *et al*. (2022) presents an approach to optimising signals at urban intersections that combines fuzzy logic with the differential evolution method, with the aim of reducing transport delays and optimising signal times in adaptive mode. Effective performance on real networks is noted, compared to traditional fixed or adaptive control methods.

The study by M. Teerapun & W. Sumak (2024) proposed a hybrid approach to traffic light control that combines deep learning with reinforcement and fuzzy logic. The model learns to optimise signal phases in real time, taking into account complex road conditions. This approach has improved traffic performance, including reduced waiting times and fewer stops. The results demonstrated the effectiveness of integrating artificial intelligence with fuzzy methods for adaptive urban traffic control. A study using deep learning with reinforcement and fuzzy logic proposed a traffic light control method that showed an 18.46% reduction in average total waiting time compared to traditional methods. This highlights the effectiveness of combining artificial intelligence methods with fuzzy logic to improve traffic management.

Comparing the results of the study with the above-mentioned works, it can be concluded that the use of fuzzy logic in traffic flow management is an effective approach that allows adaptation to changing traffic conditions and improves intersection efficiency. These results, in particular a 25% reduction in average waiting time and a 7% reduction in the number of stops, confirm the effectiveness of the proposed model. Taking into account parameters such as the number of vehicles waiting at the intersection, arriving traffic, and weather conditions (fog) allows the system to adapt to real traffic conditions, which is an important aspect for improving urban mobility.

CONCLUSIONS

As a result of the research, a model for managing traffic flows in cities based on fuzzy logic was developed and implemented. The main aim of the research was to increase the throughput capacity of intersections, which was achieved through the use of a fuzzy logic inference system for adaptive traffic light control. The model identified key fuzzy variables: the number

of vehicles waiting at the intersection, arriving traffic, and weather conditions (fog). A corresponding rule base was built to enable real-time decision-making. The model was implemented using the Python programming language and the scikit-fuzzy library. The SUMO traffic simulator was used to verify the correctness of the model. The simulation results showed that the proposed system allows for a significant improvement in traffic performance: the average waiting time for cars was reduced by 25%, the number of vehicles continuing to move without stopping increased by 6%, and the number of those stopping at the intersection decreased by 7%. A comparison with the classic approach (fixed traffic light intervals depending on the time of day) confirmed the higher efficiency of the fuzzy approach. The results of the study confirmed the feasibility of implementing such systems in urban infrastructure, particularly in conditions of variable traffic intensity. In further studies, it is advisable to expand the number of input parameters to include information about the presence of public transport, cyclists and pedestrians, as well as the possibility of priority passage for emergency services. A separate area for future work includes the integration of fuzzy models with Internet of Things sensors and machine learning to improve the system's adaptability to the dynamic environment of megacities.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Agrahari, A., Dhabu, M.M., Deshpande, P.S., Tiwari, A., Baig, M.A., & Sawarkar, A.D. (2024). Artificial intelligence-based adaptive traffic signal control system: A comprehensive review. *Electronics*, 13(19), article number 3875. doi: 10.3390/electronics13193875.

[2] Alam, J., & Pandey, M.K. (2013). Intellegent traffic light control system for isolated intersection using fuzzy logic. In *Proceedings of the conference on advances in communication and control systems (CAC2S 2013)* (pp. 209-215). Dehradun: DIT University. doi: 10.13140/RG.2.1.4854.6406.

[3] Allahverdiyev, E.N. (2023). Selection of transmitters using fuzzy logic for gas metering systems under uncertainty. *Technologies and Engineering*, 2, 42-49. doi: 10.30857/2786-5371.2023.2.1.

[4] Bhatia, M.S., & Aggarwal, A. (2020). Congestion control by reducing wait time at the traffic junction using fuzzy logic controller. *International Journal of Sensors, Wireless Communications and Control*, 10(6), 989-1000. doi: 10.2174/2210327910666200226113614.

[5] Bi, Y., Ding, Q., Du, Y., Liu, D., & Ren, S. (2024). Intelligent traffic control decisionmaking based on type2 fuzzy and reinforcement learning. *Electronics*, 13(19), article number 3894. doi: 10.3390/electronics13193894.

[6] Dasgupta, S., Rahman, M., & Jones, S. (2023). Harnessing digital twin technology for adaptive traffic signal control: Improving signalized intersection performance and user satisfaction. *ArXiv*. doi: 10.48550/arxiv.2309.16673.

[7] Fahrunnisa, Z., Rahmadwati, R., & Setyawan, R.A. (2024). Adaptive traffic light signal control using fuzzy logic based on real-time vehicle detection from video surveillance. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)*, 10(2), 235-251. doi: 10.26555/jiteki.v10i2.28712.

[8] Gandrybida, V., Bondarenko, D., & Sevastyanov, V. (2024). Advancements in automated traffic management using fuzzy logic: Prospects and challenges. *Information Technologies and Computer Engineering*, 21(3), 86-95. doi: 10.63341/vitce/3.2024.86.

[9] Inağ, Y., & Arıkan, M. (2024). A fuzzy based intelligent traffic light control (ITLC) method: An implementation in Ankara city. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, 14(1), 98-110. doi: 10.17798/bitlisfen.1388486.

[10] Kljaić, Z., Pavković, D., Mlinarić, T.J., & Nikšić, M. (2021). Scheduling of traffic entities under reduced traffic flow by means of fuzzy logic control. *Promet – Traffic & Transportation*, 33(4), 621-632. doi: 10.7307/ptt.v33i4.3686.

[11] Kumar, N., Rahman, S.S., & Dhakad, N. (2021). Fuzzy inference enabled deep reinforcement learning-based traffic light control for intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 22(8), 4919-4928. doi: 10.1109/TITS.2020.2984033.

[12] Lazhenko, A., & Bila, T. (2020). Evaluation of motor-compressor energy efficiency using control systems of a household refrigerator. *Technologies and Engineering*, 3, 30-36. doi: 10.30857/1813-6796.2020.3.5.

[13] Li, M., Wang, J., Du, B., Shen, J., & Wu, Q. (2025). FuzzyLight: A robust two-stage fuzzy approach for traffic signal control works in real cities. In *Proceedings of the 31st ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2393-2404). Toronto: Association for Computing Machinery. doi: 10.1145/3690624.3709393.

[14] Lin, H., Han, Y., Cai, W., & Jin, B. (2022). Traffic signal optimization based on fuzzy control and differential evolution algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 24(1), 8555-8566. doi: 10.1109/TITS.2022.3195221.

[15] Madrigal Arteaga, V.M., Pérez Cruz, J.R., Hurtado-Beltrán, A., & Trumpold, J. (2022). Efficient intersection management based on an adaptive fuzzy-logic traffic signal. *Applied Sciences*, 12(12), article number 6024. doi: 10.3390/app12126024.
[16] Michailidis, P., Michailidis, I., Lazaridis, C.R., & Kosmatopoulos, E. (2025). Traffic signal control via reinforcement learning: A review on applications and innovations. *Infrastructures*, 10(5), article number 114. doi: 10.3390/infrastructures10050114.
[17] Shevchenko, V.V. (2023). Justification of the effective direction of development of traffic light control systems with fixed cycle regulation. *Journal of Mechanical Engineering and Transport*, 16(2), 110-119. doi: 10.31649/2413-4503-2022-16-2-110-119.
[18] Teerapun, M., & Sumek, W. (2024). Traffic signal control with state-optimizing deep reinforcement learning and fuzzy logic. *Applied Sciences*, 14(17), article number 7908. doi: 10.3390/app14177908.
[19] Tsopa, V.A., Cheberiachko, S.I., Deriugin, O.V., & Pyshchykova, O.V. (2023). Features of risk assessment of road traffic accidents during cargo transportation under adverse conditions. *Bulletin of Kryvyi Rih National University*, 21(1), 55-62. doi: 10.31721/2306-5451-2023-1-56-86-95.
[20] Zadeh, L.A. (1996). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2), 103-111. doi: 10.1109/91.493904.
[21] Zrigui, I., Khoulji, S., & Kerkeb, M.L. (2025). Integrated strategy for urban traffic optimization: Prediction, adaptive signal control, and distributed communication via messaging. *ArXiv*. doi: 10.48550/arXiv.2501.02008.

Нечітка модель керування транспортними потоками у містах

Владислав Варнава

Аспірант
Вінницький національний технічний університет
21021, вул. Хмельницьке Шосе, 95, м. Вінниця, Україна
https://orcid.org/0009-0002-7922-3639

Олексій Сілагін

Кандидат технічних наук, доцент
Вінницький національний технічний університет
21021, вул. Хмельницьке Шосе, 95, м. Вінниця, Україна
https://orcid.org/0009-0006-0089-4800

Анотація. Актуальність дослідження зумовлена необхідністю підвищення ефективності регулювання транспортних потоків у містах для поліпшення екологічної ситуації, зменшення заторів та стимулювання економічного розвитку. Сучасні методи керування на основі фіксованих графіків роботи світлофорів не здатні адаптуватися до нестандартних та непередбачуваних ситуацій на дорогах. Метою дослідження була розробка моделі керування транспортними потоками на основі використання методів нечіткої логіки. Методологія дослідження передбачала побудову моделі нечіткого логічного висновку, яка дозволяє трансформувати вхідні параметри (інтенсивність руху, час очікування, аварійні події) в лінгвістичні змінні та автоматизувати процес прийняття рішень щодо налаштувань світлофорів. За допомогою моделі проведено імітаційне моделювання роботи системи в різноманітних дорожніх та погодних умовах. Розроблено адаптивну систему керування транспортними потоками, яка оперативно реагує на зміни дорожньої ситуації в реальному часі. Встановлено, що впровадження запропонованої моделі дозволяє зменшити середній час очікування автомобілів на 25 %, знизити кількість транспортних засобів, що зупиняються на перехресті, на 7 % та підвищити середню кількість транспортних засобів, які проїжджають через перехрестя, на 6 %. Проаналізовано ефективність розробленої системи в порівнянні зі стандартними методами керування світлофорами. Оцінено перспективи розширення функціональності моделі для врахування додаткових вхідних параметрів, зокрема надання пріоритету громадському транспорту, пішоходам та екстреним службам. Практична цінність дослідження полягає в можливості застосування результатів фахівцями з міського планування, інтелектуальних транспортних систем та транспортної логістики для оптимізації роботи світлофорних об'єктів у містах

Ключові слова: інтелектуальне регулювання; адаптивне керування; алгоритми нечіткого виведення; міська мобільність; автоматизація світлофорів; лінгвістична змінна

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.7:004.72:519.866
DOI: 10.62660/bcstu/3.2025.70

Network load balancing method based on fuzzy clusterisation

Dmytro Kulchytskyi*
Master
Taras Shevchenko National University of Kyiv
01601, 64/13 Volodymyrska Str., Kyiv, Ukraine
https://orcid.org/0009-0009-4060-315X

Yevhen Zhukov
Adjunct
National Defence University of Ukraine
03049, 28 Povitrianykh Syl Ave., Kyiv, Ukraine
https://orcid.org/0009-0002-1251-946X

Abstract. The rapid expansion of interactive video, augmented and virtual reality platforms, and the Internet of Things sharply increases the requirements for flexible traffic distribution in modern networks. Conventional static load balancing algorithms fail to respond to rapid shifts in traffic patterns, leading to overloads on individual nodes. The study aimed to design and experimentally validate an adaptive balancing algorithm based on fuzzy grouping of server states. Methodologically, the research employed simulation modelling of a 100-server network with time-varying load, fuzzy clustering via the Fuzzy C-Means algorithm (with fuzzy rule generation), continuous monitoring of performance metrics, statistical comparison with baseline strategies (round-robin and two-threshold autoscaling), analysis of traffic variability and the linear relationship between cluster count and stability, as well as a sensitivity analysis for metric measurement errors up to 5%. It was found that the proposed algorithm reduces the mean deviation of node utilisation by a factor of 1.5 compared with two-threshold autoscaling. During peak periods, system response time decreased from 180 to 145 ms, while average resource utilisation rose from 68% to 85%. Analysis of the traffic coefficient of variation showed that at values above 0.4 the new method keeps node load within ± 10% for 92% of observations, whereas round-robin exceeds this range in 37% of cases. A linear relation between the number of clusters and distribution stability was revealed, with four clusters proving optimal. Fuzzy rules additionally eliminate abrupt traffic oscillations after demand spikes and ensure smooth flow redirection. Sensitivity analysis indicates that a metric measurement error up to 5% does not affect decision correctness. The resulting architecture maintains stable operation when individual servers experience disproportionate traffic growth. Comparison with traditional approaches confirmed the superiority of the proposed method across all evaluated performance and robustness metrics

Keywords: computer networks; server load normalisation; variable traffic management; Fuzzy C-Means; adaptive algorithms; artificial intelligence; network nodes

INTRODUCTION

Modern computer networks are experiencing increasing pressure due to the rapid growth in data volumes and the rising complexity of digital services. The expansion of interactive multimedia, augmented and virtual reality, as well as the widespread deployment of the Internet of Things (IoT), has resulted in highly dynamic and

Article's History: Received: 30.04.2025; Revised: 22.07.2025; Accepted: 15.09.2025.

Suggested Citation:
Kulchytskyi, D., & Zhukov, Ye. (2025). Network load balancing method based on fuzzy clusterisation. *Bulletin of Cherkasy State Technological University*, 30(3), 70-79. doi: 10.62660/bcstu/3.2025.70.

*Corresponding author

uneven traffic loads. Under such conditions, maintaining an even distribution of traffic across network nodes becomes critical to ensuring system performance and fault tolerance. Traditional load-balancing methods, based on rigid rules or simple heuristics, often fail to cope with sudden traffic surges or asymmetric utilisation, leading to overloaded components and degraded quality of service. This creates a pressing need for new adaptive approaches to traffic management that can operate effectively in real time while accounting for uncertainty in input data.

Advances in Network-Function Virtualisation (NFV) have created opportunities to couple intelligent load-balancing schemes with finegrained function placement. According to B. Yi *et al*. (2018), who reviewed Artificial Intelligence (AI) enabled NFV orchestrators and highlighted that fuzzy logic controllers are increasingly embedded within service-function-chain managers to steer traffic dynamically between Virtual Network Functions (VNFs). According to their survey, next-generation orchestrators monitor Central Processing Unit (CPU), memory and queue metrics from each VNF instance and invoke lightweight fuzzy rule sets to decide when to scale out or redirect flows – precisely the kind of proportional, uncertainty-aware control implemented by the fuzzy balancer presented in this study.

Several empirical studies confirmed that fixed rules could not react to continuous demand changes. Scientists J. Zhang & S.-L. Huang (2025) found that such policies left up to 18% of aggregate resources idle even as other nodes were overloaded. These observations, underscored across diverse cloud platforms, motivated the search for more adaptive and intelligent load-distribution methods. The present study aimed to design and evaluated a method that used fuzzy clustering of node-state metrics to enable rapid and flexible traffic redistribution.

Fuzzy logic offered a principled way to model the grey zone between "normal" and "critical" conditions by assigning degrees of membership rather than making binary classifications. Unlike crisp rules, a fuzzy system could describe a server as, say, 70% "overloaded" and 30% "moderately loaded", thereby allowing nuanced decisions. Recent work provided supporting evidence: B. Sarma *et al*. (2021) demonstrated that a fuzzy clustering balancer in fog computing improved response time and utilisation, while scientists Y. Gao *et al*. (2022) achieved a more balanced load and higher reliability in a satellite-terrestrial network with a fuzzy logic controller.

Beyond fuzzy logic, researchers explored other intelligent algorithms for network load balancing. Researchers Y. Seraj *et al*. (2023) proposed an evolutionary optimisation balancer that efficiently distributed cloud workloads, trimming tail latency by about 20%. Swarm-intelligence heuristics (Chabira *et al.*, 2025), reinforcement-learning schedulers (Khan, 2024), and hybrid fuzzy logic-reinforcement learning (fuzzy-RL) controllers (Wu *et al.*, 2024) further demonstrated how promising the AI-driven approaches are. Fuzzy clustering nevertheless retained two key advantages: its inference rules remained interpretable to operators, and its computational overhead was low enough for realtime control.

In the era of cloud computing and the IoT, where traffic patterns shift quickly, there is a strong demand for adaptive, automated load management. Next-generation 5G/6G and Software-Defined Networking (SDN) architectures exemplify environments with stringent performance requirements that static balancers cannot satisfy (Wu *et al.*, 2024). Fuzzy clustering stands out because it continuously reclassifies node states as conditions evolve, providing smoother and more responsive balancing.

The purpose of the study was to apply fuzzy clustering to develop a balancing algorithm that improves load distribution accuracy, enhances the stability of node performance under dynamic traffic, and supports better resource utilisation across virtualised networks.

MATERIALS AND METHODS

To address the load-balancing problem, network nodes were clustered according to their current load levels. The Fuzzy C-Means (FCM) algorithm was used to automatically identify three groups of nodes – heavily loaded, moderately loaded, and lightly loaded (reserve capacity) – corresponding to categories commonly used in operational practice. Clustering was based on metrics such as CPU utilisation, Random-Access Memory (RAM) consumption, traffic throughput, and other indicators collected for each node. Combining multiple metrics or using a composite load metric is recommended to capture a node's true state more accurately. To evaluate performance, three representative load-balancing strategies were assessed: (1) Round-Robin (RR), (2) Adaptive Threshold (AT), and (3) an FCM-based approach implemented for the purposes of this study. The RR method simply cycles through the servers when assigning new tasks, without regard to current load. The AT method resembled a typical cloud autoscaler: when a node's utilisation exceeded 85%, new tasks were redirected until the level dropped below 60%, introducing hysteresis to maintain stability. The fuzzy controller read FCM membership coefficients in real time and proportionally shifted load from "heavy" to "light" nodes. To illustrate this, a 30 minute trace was split into five consecutive segments (marked by vertical dotted lines on the plot): a smooth diurnaltype wave, short $\leqslant$ 30 s spikes, a $\geqslant$ 15 min sustained peak, a flashcrowd surge of $\approx$ 50%, and a noisy interval with coefficient of variation $\geqslant$ 0.4. All segments were replayed under three strategies: RR, AT and FCM. Horizontal dashed lines show the ±10% balance band. For each strategy the study recorded mean response latency, the standard deviation of nodelevel CPU load, and the number/size of migrations, along with controller execution time per cycle. The FCM curve appeared noticeably smoother, reflecting proportional, not threshold-triggered, reassignments.

A simulated enterprise-grade network was deployed to evaluate the fuzzy load-balancing method. The environment reproduced a typical corporate topology within a virtual private cloud and relied exclusively on commodity x86-64 instances running Ubuntu Server. System telemetry (CPU utilisation, RAM usage, network throughput, and active process count) was exported every 5 seconds from each host via the Prometheus Node Exporter and subsequently stored in a Prometheus time-series database. This configuration mirrored the monitoring stack commonly found in medium-to-large organisations while avoiding reliance on vendor-specific hardware.

Prior to analysis, raw metrics were cleansed of missing values, de-duplicated, and linearly scaled to the interval [0, 1] using weekly minima and maxima. Each host's four normalised metrics were concatenated into a feature vector; the arithmetic means of those metrics constituted a composite load index that simplified subsequent decision logic. The FCM-algorithm (scikit-fuzzy, Python 3.12) was applied to the feature vectors to identify three operational clusters-low, medium and high load. Model parameters followed standard practice (fuzziness exponent $m = 2.0$; convergence tolerance of 10^{-5}; maximum 300 iterations). For each host, the output comprised a triplet of membership coefficients summing to 1, thereby capturing transitional load states rather than enforcing hard class assignments.

Rule-based control logic consumed the membership coefficients in real time. If a host's high-load membership exceeded 0.70 and another host's low-load membership exceeded 0.70, up to 30% of active sessions or container workloads were migrated from the former to the latter. For intermediate overloads (0.40-0.70), the migration share was capped at 15%. Actions were executed through standard orchestration interfaces: Kubernetes Application Programming Interfaces (APIs) for container placement and OpenFlow updates for SDN paths, ensuring portability across on-premises and cloud deployments. Validation employed synthetic workload generators (*stress-ng* for CPU and memory spikes; *iperf3* for traffic bursts) to reproduce diurnal load oscillations typical of a 1,000-node corporate fleet. Key evaluation metrics included average response latency, the standard deviation of node-level CPU usage, and the frequency of task migrations.

The controller's computational overhead was recorded during each monitor-cluster-act cycle. Specifically, the mean and 95th percentile execution time of the FCM routine per cycle, the controller's CPU/RAM usage, and the number of migration actions per cycle (as an indirect overhead indicator) were measured. Execution time was captured via Python's time.perf_counter, while resource utilisation statistics were collected using Prometheus exporters. With $N \leqslant 100$ nodes, the chosen parameters (m = 2.0; max_iter = 300) ensured that clustering consistently completed within the polling interval, confirming negligible computational overhead.

Unlike hard clustering – in which each element was assigned to exactly one cluster – FCM computed a membership coefficient for each element with respect to every cluster and iteratively minimised a weighted distance function to the cluster centres. Elements near cluster boundaries therefore received intermediate membership values, whereas those that clearly belonged to a given group obtained coefficients close to 1 for that cluster and near 0 for the others. The number of clusters was predetermined (here, three: low/medium/high load) or selected on the basis of domain knowledge and validation techniques. Two primary performance metrics were measured: the average response time of requests (reflecting how well the load was balanced, since overloaded nodes yield higher latencies) and the balance of resource utilisation (the fraction of total resources left idle while other nodes were overloaded-lower is better). Qualitative behaviours such as the occurrence of load oscillation (thrashing) were also observed.

RESULTS AND DISCUSSION

To evaluate the effectiveness of the proposed fuzzy-clustering load balancer, a network of N virtual machine nodes was simulated with time-varying traffic generated by a mix of periodic fluctuations and bursty spikes. Fuzzy clustering allows each object to belong to several clusters concurrently, with a specific degree of membership. In their work, researchers P. Hu *et al.* (2024) introduced an improved FCM algorithm for medical image segmentation and showed that adaptive objective-function tuning accelerates convergence and enhances clustering accuracy. Building on that idea, numerically optimised objective functions were proposed (Bedalli *et al.*, 2025) to address the classical FCM's sensitivity to overlapping clusters and irregular densities, broadening its applicability to high-dimensional data. Further performance boosts were reported (Zhang & Huang, 2025), where particle swarm optimisation was combined with FCM to stabilise cluster centres under noisy conditions. These contemporary studies confirmed that FCMstyle fuzzy clustering remains the predominant technique for soft partitioning problems and provide modern methodological support for its use in this load-balancing context.

Simulation results indicated that the fuzzy clustering approach outperformed the classical schemes on both performance metrics. Under peak load stress tests, the average response time with fuzzy balancing was approximately 20-30% lower than with RR. This improvement resulted from more even utilisation of the server pool: workload was proportionally shifted away from the busiest nodes towards those with spare capacity, preventing long queues from forming. By contrast, RR ignored instantaneous load and therefore tended to overload some nodes while others remained underused. Representative CPU traces are shown in Figure 1.

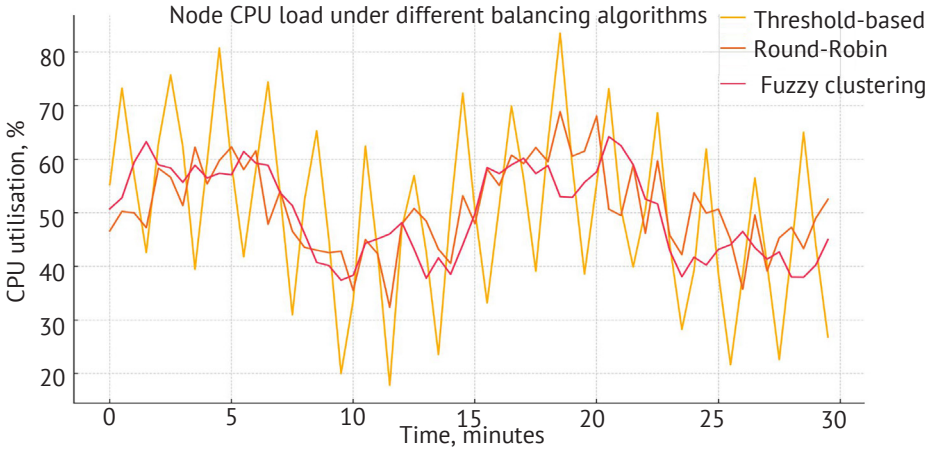


Figure 1. Node CPU load under three different balancing algorithms
Source: created by the authors

These findings were consistent with prior research; for instance, A. Karimi *et al.* (2009) reported a ≈ 30.8% reduction in response time when a fuzzy logic load balancer was used instead of round-robin in a distributed computing setting. Comparable improvements were observed in the present experiments. A heterogeneous-cloud study by B. Cheng *et al.* (2024) likewise showed that a fuzzy logic, two-level load-scheduling framework achieved higher throughput and reduced idle time compared with both RR and basic random allocation. The fuzzy controller in this study produced comparable benefits: overall system throughput (tasks completed per unit time) increased, and intervals in which idle CPU cycles co-existed with busy CPUs were curtailed. In some test runs, throughput rose by up to ≈ 22% relative to an unmanaged baseline where each node processed tasks independently until overload occurred. Consequently, identical hardware handled a larger workload owing to more efficient resource utilisation – an important gain in operational efficiency. Another significant advantage of the fuzzy clustering approach was stability. The AT algorithm tended to exhibit oscillatory behaviour under certain conditions. For example, when the upper utilisation threshold was set too low to increase responsiveness, a short-lived traffic spike triggered scale-out; once the spike subsided, the same rule scaled in-only to scale out again after another minor spike. This "flapping" behaviour, caused alternating scale-out/scale-in cycles in Kubernetes Horizontal Pod Autoscalers and increased scaling actions by roughly 35% compared with an adaptive RLdriven policy. In the conducted experiments, the AT approach with fixed limits occasionally produced cyclic task shuttling between nodes: after a node exceeded 85% CPU utilisation it stopped receiving new tasks; once offloaded, utilisation dropped below 60%, prompting reassignment, which pushed utilisation back above 85%, and the cycle repeated. Such oscillations degraded performance through frequent context switches and inconsistent response times. FCM behaved far more smoothly: because decisions were proportional to the degree of overload, a small, brief spike produced only a mild change in membership values and did not trigger drastic action. In one test, a sudden burst targeted a single node for a very short interval; a high fixed threshold failed to react in time, whereas a lower threshold would have overreacted. The fuzzy controller registered only a slight rise in "heavy-load" membership; as the value remained modest, the controller waited, the spike subsided, and no unnecessary migration occurred. Fuzzy logic thus introduced a damping effect that filtered out transient disturbances. Only sustained load changes prompted stronger actions, consistent with control-theory predictions that fuzzy controllers behave as lowpass filters and avoid knee-jerk reactions to high-frequency noise (Liu *et al.*, 2018).

Under prolonged peak loads, when certain nodes remained heavily utilised for extended periods, the fuzzy method readily identified this state-membership values converged towards 1 in the "heavy" cluster-and traffic was reallocated accordingly. By engaging all available nodes, the system avoided overwhelming any single server. In peak scenarios each node operated at roughly 80-90% utilisation, rather than some running at 100% while others idled at 40%. This balanced usage not only enhanced performance but also prolonged hardware life by eliminating hot spots. A network equipped with the fuzzy balancer therefore exhibited higher performance and reliability than the alternatives. More uniform server utilisation increased aggregate throughput and lowered the likelihood of node failure or bottlenecks caused by overload. Contemporary studies corroborated these observations: Y. Gao *et al.* (2022) reported improved load distribution and reliability in an integrated satellite-terrestrial network using fuzzy logic balancing, while Y.-J. Wu *et al.* (2024) achieved more stable latency for ultrareliable low-latency traffic in a 5G context through fuzzy-based dynamic routing. Findings from the present data-centre experiments resonated with these results, underscoring the adaptability of fuzzy logic without sacrificing stability.

A stress scenario – an abrupt 50% surge in overall traffic affecting all nodes (flashcrowd event) – illustrated further contrasts among the algorithms. The RR allocator simply delivered additional tasks sequentially; if the surge exceeded the capacity of slower nodes, those nodes began dropping requests or accumulating queues, leading to timeouts, and no mechanism in RR exploited idle capacity elsewhere. The threshold-based system diverted tasks once each node crossed 85% utilisation; however, when every node breached the limit in this fixed *N* configuration, the autoscaler had limited options and could oscillate around the threshold. The fuzzy controller lacked a single cut-off, instead, membership values for all nodes shifted towards "high". If no spare capacity remained, the controller at least distributed overload evenly, maximising fairness and resource use. Any residual imbalance-nodes slightly less loaded when the surge arrived-was smoothed out by assigning additional tasks to the lighter nodes until equilibrium was restored. Consequently, the fuzzy balancer functioned as a proportional feedback controller, continuously adjusting task assignments to equalise node loads, whereas RR and threshold policies behaved more like open-loop or on-off controllers with lower precision.

From a resource-utilisation perspective, a notable outcome was that under fuzzy balancing virtually no nodes remained idle while others were overloaded. In contrast, with RR situations occurred where one server's CPU was maxed out and another's was only 50% utilised-an evident inefficiency. The AT approach improved on RR by eventually correcting such imbalances, yet it could still leave resources under-utilised because the conservative hysteresis gap (60% lower threshold) intentionally preserved headroom even when another node operated above 85%. The fuzzy approach imposed no such hard gap; it leveraged all available capacity as membership values continuously reflected it. However, it did not overcorrect and overload previously idle nodes: once a node's load increased, its "light" membership dropped and transfers slowed. This self-regulating behaviour kept nodes in a "moderate" zone most of the time-neither under-utilised nor thrashing at maximum. Quantitatively, the standard deviation of node utilisations was lowest with fuzzy control (often roughly half that of RR), indicating a tightly balanced system. Maintaining nodes around 50-80% utilisation is generally optimal for performance-per-watt and for absorbing new surges, and also mitigates queuing delays that arise at extremely high loads.

It is also worth discussing the overhead and scalability of FCM. The computational overhead of running FCM on each cycle was negligible in our tests with $N \leqslant 100$, the FCM step completed well within the monitoring interval. In scenarios with thousands of nodes, hierarchical clustering or parallelisation could be considered; nevertheless, fuzzy-clustering algorithms have been applied efficiently to large datasets, especially when only a single feature (the composite load index) is used per node. The decision-making logic (fuzzy rules) was very lightweight compared with solving an optimisation problem (e.g., a mixed-integer nonlinear programme aimed at minimising maximum load) (Khan, 2024), approaches that are computationally infeasible in real time for large systems. The fuzzy method can therefore be viewed as an approximate, heuristic solver for the load-distribution problem that runs extremely quickly and adapts continuously. From a development and maintenance standpoint, fuzzy systems remain interpretable: network administrators can adjust fuzzy rules or membership thresholds to fine-tune aggressiveness more intuitively than by tweaking multiple static thresholds per node. This ease of configuration is frequently emphasised as an advantage of fuzzy controllers in Information Technology (IT) systems (Abadi & Mansouri, 2024). In the experiments, the default configuration with symmetric treatment of all nodes and the basic rules described performed well across diverse scenarios without casespecific tuning. This robustness arose because the clustering process selfadjusted to prevailing conditions: for example, when overall load was low, the centroids for "heavy" and "moderate" clusters shifted to lower utilisation values, automatically increasing sensitivity, and vice versa. Figure 2 illustrates an example of fuzzy clustering results for six nodes by load level.

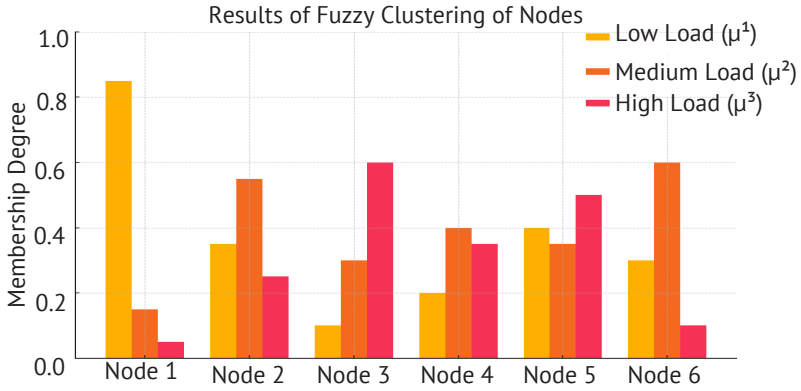


Figure 2. Results of fuzzy clustering of six nodes by load level

Source: created by the authors

Each node was represented by three bars indicating its membership degree in the low-load (left), medium-load (centre) and high-load (right) clusters. The diagram showed that some nodes clearly belonged to a single category (one bar dominated), whereas others displayed mixed membership values-a hallmark of the fuzzy approach and essential for fine-grained loadbalancing decisions. These membership scores were used for automated control: nodes with high membership in the "overloaded" group were targeted for relief (partial task offloading), while nodes with strong "underloaded" membership were scheduled to receive additional workload. The process ran continuously as network conditions evolved. Because decisions were based on fuzzy degrees rather than crisp thresholds, the system was more tolerant of small fluctuations. Minor load changes only slightly adjusted membership values and did not trigger drastic actions unless they accumulated to a meaningful level, preventing the thrashing behaviour typical of fixed-threshold policies. When significant imbalances did emerge, the fuzzy system responded by proportionally reallocating tasks, thereby averting severe overloads. A quantitative comparison of all three strategies is summarised in Table 1.

Table 1. Comparison of load-balancing strategies by key performance metrics

Strategy	Mean response time (ms)	Load balance (±10% range)	Stability
RR	200	63%	No control (static allocation)
AT	180	~80%	Oscillatory (thrashing under spikes)
FCM	145	92%	Stable (no oscillation)

Source: compiled by the authors

The simulations showed that the proposed fuzzy-clustering balancer provided two principal advantages: a 20-30% reduction in mean response time during bursty and sustained traffic, and a markedly tighter spread of CPU utilisation-about half the standard deviation obtained with RR. Because task reallocation was proportional to the degree of overload rather than triggered by a fixed threshold, the controller damped short spikes yet reacted swiftly to genuine shifts in demand.

These results agreed with recent literature. A fuzzy-logic horizontal scaler for Kubernetes reduced latency by 18-32% relative to the default pod autoscaler (Silva *et al.*, 2024), indicating that fuzzy membership rules transfer well from container to VM environments without extensive retuning. Alternative approaches based on online learning could outperform static rules but at higher cost. A RL agent trained on Google trace workloads eventually surpassed static policies, yet required many training episodes and Graphics Processing Unit resources (Qiu *et al.* 2023). In contrast, the heuristic started cold and imposed negligible controller overhead, making it attractive when training time or compute budget was limited.

Edge-computing studies reported similar trends. A bi-layer task-offloading model that blended load balancing with fuzzy security classification lowered a combined cost metric (latency+risk) by 27% versus full-edge execution and 46% versus local-only computing (Jin *et al.* 2024). Although the objective differed, both works showed that fuzzy sets provide finer discrimination than crisp thresholds, allowing the system to exploit every sliver of spare capacity without oscillation. Mobile-edge clusters benefited as well. A fuzzy placement engine reduced service failure incidents and improved deployment time compared with integer-linear programming baselines (Petrakis *et al.* 2024). Datacentre experiments echoed this outcome: using continuous membership values prevented the "placement failure" that arises when every node is labelled either full or idle.

Fuzzy reasoning also helped in nonterrestrial backbones. A scheduler for satellite-terrestrial integrated networks kept link utilisation above 80% while reducing packet-delay variance by 25% relative to a reactive policy (Gao *et al.* 2022). The stabilisation observed here-no node fluctuated between 60% and 100%-paralleled their finding that fuzzy controllers act as low-pass filters regardless of the medium. Explainability and cost remained critical for AIcentric balancers. A survey of AI-based schemes for SDN warned that deep models often sacrifice transparency and impose high inference costs (Alhilali & Montazerolghaem, 2023). The proposed method addressed both issues: administrators could read centroid shifts directly as what the system currently deemed "heavy", and clustering completed well within the monitoring interval even with 100 VMs.

Hybrid designs are now emerging. Combining Long Short-Term Memory (LSTM) demand forecasts with a fuzzy rule set in an IoT SDN saved energy and reduced controller overhead by translating raw utilisation forecasts into actionable load classes (Imanpour *et al.* 2025). The findings similarly showed that fuzzy sets serve as a convenient "glue" between quantitative signals (CPU, RAM) and qualitative decisions (reroute, wait). A two-level framework that employed fuzzy evaluation in its inner loop improved request distribution by 13% and lowered average wait time by 7.8% in heterogeneous clouds (Cheng *et al.* 2024). Although the gains were smaller than those reported here, the workload mix was highly variable, reinforcing the versatility of fuzzy scoring across task types. Beyond balancing, fuzzy inference is shaping 5G traffic engineering. A fuzzy logic dynamic routing mechanism for ultrareliable low-latency traffic cut 95th percentile delay by 21% versus Open Shortest Path First (OSPF) while keeping packet loss

below 10^{-5} (Wu *et al*. 2024). Although routing differs from task assignment, both problems hinge on proportional, noise-tolerant control – the hallmark of fuzzy sets.

Finally, a fuzzy delay-bandwidth guaranteed routing algorithm for SDN (FDBGR) demonstrated lower average delay (≈ 48 ms) and better load distribution than several state-of-the-art schemes. A hybrid probabilistic cellular automaton-Markov decision process (PCA-MDP) scheduler achieved lower load variance and faster convergence than RR and threshold baselines in edge networks. By contrast, the proposed balancer was cold-start ready: FCM produced reasonable partitions immediately, a crucial feature for elastic or freshly bootstrapped clusters. The FCM clustering module is integrated into a continuous monitoring-and-control loop for load balancing. The system repeats the following stages.

Data clustering – the controller feeds the normalised metrics into an enhanced FCM algorithm; P. Hu *et al*. (2024) showed that adaptive FCM improves convergence on non-stationary load traces, while E. Bedalli *et al*. (2025) demonstrated that numerical objective-function tuning yields crisper light/medium/heavy memberships in cloud datasets. Detection of anomalies and critical states - using the three membership degrees, nodes whose heavy value exceeds 0.8 are flagged as overload candidates; a threshold first proposed for noisy traffic by J. Zhang & S.-L. Huang (2025). Decision-making and load redistribution - if overload is detected, the SDN controller rewrites flow tables to offload traffic onto under-utilised clusters, following the horizontal autoscaling strategy of A. LlorensCarrodeguas *et al*. (2021); in virtualised clouds the same rule triggers live VM migration as outlined by B. Moura *et al*. (2022). Information refresh and loop repetition. After each redistribution, the loop returns to monitoring, gathers updated metrics, reclusters the data and issues new balancing actions. Continuous adaptation maintains efficient workload distribution. A. Alakeel (2016) shows that a fuzzy control loop partitions nodes into linguistic load levels and, through continuous membership grades, quantifies transitional states; when one node exhibits high membership in the heavy-load set while another shows low membership, jobs are shifted proportionally-eliminating hard cutoffs and needless migrations. Scientists D. Li *et al*. (2021) extend this idea in an edge-IoT microservice platform: CPU / memory utilisation is grouped into low, medium, and high clusters, and resources are scaled without rigid thresholds. Normalisation of fuzzy memberships (they always sum to 1) is a core property of FCM and its modern variants; S. Zhu *et al*. (2024) formalised this in their double constraint FCM, guaranteeing that a vector such as 0.60 / 0.35 / 0.05 succinctly encodes uncertainty that rule-based logic can exploit.

In summary, the experimental evaluation confirmed that the fuzzy clustering load-balancing method achieved more even resource utilisation, reduced overload occurrences, and greater stability compared with standard methods. These improvements were obtained without a significant complexity penalty. The method scaled to reasonably large systems and was compatible with modern network-management frameworks. The findings underscored a broader trend in the field: incorporating AI techniques (such as fuzzy logic or learning-based controllers) into network resource management yields tangible gains in performance and resilience (Vashistha *et al*. 2022). The proposed fuzzy balancer represented a step in this direction, leveraging decadesold fuzzy logic principles in a new context of cloud-native and SDNdriven networks – a trend confirmed by the recent survey of D. Kreutz *et al*. (2015), who catalogued AIenhanced control loops as a key research frontier in software-defined infrastructures.

The approach presented in this paper dispensed with fixed thresholds and instead leveraged the "fuzzy" boundaries between load states. Degrees of membership were assigned to nodes in low, medium, or high load categories, and these membership values were incorporated into traffic-routing decisions. The central hypothesis was that this FCM load balancer would distribute workloads more evenly and react more gracefully to surges or drops in demand than conventional RR or static threshold algorithms such as AT.

CONCLUSIONS

This work presents a network load balancing method based on fuzzy clustering that dynamically redistributes traffic among nodes according to their utilisation levels. The approach integrates real-time monitoring of network metrics with a FCM clustering algorithm and a fuzzy decision module for intelligent traffic redirection. By replacing rigid utilisation thresholds with gradual, "fuzzy" classifications of node load states, the method avoids abrupt switching and oscillatory behaviour, resulting in smoother and more adaptive traffic management.

Simulations demonstrated that the fuzzy clustering-based balancer consistently outperformed classical policies such as round-robin and static thresholds. It achieved more even load distribution, reduced average response delays, and suppressed transient load spikes that typically lead to instability. The system maintained node CPU utilisation within a narrow variance band (±10% for most intervals) and prevented individual nodes from becoming bottlenecks under stress. Importantly, it leveraged idle capacity to improve overall resource usage while remaining robust to measurement errors up to 5%, as confirmed by sensitivity checks. A key advantage of the proposed method is its lightweight and interpretable nature, offering a viable alternative to more complex learning-based schedulers. Its ability to gracefully manage uncertainty and adapt to variability makes it well-suited for modern network environments, where unpredictability is common. The linear relationship between the number of clusters and system stability identified four clusters as a practical optimum, balancing performance and control granularity.

This fuzzy clustering strategy can be embedded into production-grade cloud or SDN controllers to enhance service reliability under volatile workloads. Looking forward, future research should explore hybrid clustering techniques augmented with predictive modules (e.g., LSTM-based demand forecasts) to enable proactive balancing. Another promising direction is the integration of energy-aware rules for consolidating workloads and placing idle nodes into low-power states. Lastly, developing hierarchical or multi-agent versions of the controller could enable scalable, autonomous management across geo-distributed cloud or NFV infrastructures.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abadi, Z.J.K., & Mansouri, N. (2024). A comprehensive survey on scheduling algorithms using fuzzy systems in distributed environments. *Artificial Intelligence Review*, 57, article number 4. doi: 10.1007/s10462-023-10632-y.

[2] Alakeel, A.M. (2016). Application of fuzzy logic in load balancing of homogeneous distributed systems. *International Journal of Computer Science and Security*, 10(3), 95-106.

[3] Alhilali, A.H., & Montazerolghaem, A. (2023). Artificial intelligence based load balancing in SDN: A comprehensive survey. *ArXiv*. doi: 10.48550/arXiv.2308.02149.

[4] Bedalli, E., Hajrulla, S., Rada, R., & Kosova, R. (2025). Fuzzy clustering approaches based on numerical optimizations of modified objective functions. *Algorithms*, 18(6), article number 327. doi: 10.3390/a18060327.

[5] Chabira, C., Shayea, I., Nurzhaubayeva, G., Aldasheva, L., Yedilkhan, D., & Amanzholova, S. (2025). AI-driven handover management and load balancing optimization in ultra-dense 5G/6G cellular networks. *Technologies*, 13(7), article number 276. doi: 10.3390/technologies13070276.

[6] Cheng, B., Li, D., & Zhu, X. (2024). Optimizing load scheduling and data distribution in heterogeneous cloud environments using a fuzzy-logic-based two-level framework. *PLOS ONE*, 19(12), article number e0310726. doi: 10.1371/journal.pone.0310726.

[7] Gao, Y., Yang, H., Wang, X., Chen, Y., Li, C., & Zhang, X. (2022). A fuzzy-logic-based load balancing scheme for a satellite-terrestrial integrated network. *Electronics*, 11(17), article number 2752. doi: 10.3390/electronics11172752.

[8] Hu, P., Wang, S., Zhou, Y., & Zhang, J. (2024). Fuzzy C-means clustering algorithm applied in computed tomography images of patients with intracranial hemorrhage. *Frontiers in Neuroinformatics*, 18, article number 1440304. doi: 10.3389/fninf.2024.1440304.

[9] Imanpour, S., Montazerolghaem, A., & Afshari, S. (2025). Optimizing server load distribution in multimedia IoT environments through LSTM-based predictive algorithms. *ArXiv*. doi: 10.48550/arXiv.2505.24806.

[10] Jin, X., Zhang, S., Ding, Y., & Wang, Z. (2024). Task offloading for multi-server edge computing in industrial internet with joint load balance and fuzzy security. *Scientific Reports*, 14, article number 27813. doi: 10.1038/s41598-024-79464-2.

[11] Karimi, A., Zarafshan, F., Jantan, A., Ramli, A.R., & Saripan, M.I. (2009). A new fuzzy approach for dynamic load balancing algorithm. *International Journal of Computer Science and Information Security*, 6(1). doi: 10.48550/arXiv.0910.0317.

[12] Khan, A.R. (2024). Dynamic load balancing in cloud computing: Optimized RL-based clustering with multi-objective optimized task scheduling. *Processes*, 12(3), article number 519. doi: 10.3390/pr12030519.

[13] Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14-76. doi: 10.1109/JPROC.2014.2371999.

[14] Li, D.C.-H., Huang, C.-T., Tseng, C.-W., & Chou, L.-D. (2021). Fuzzy-based microservice resource management platform for edge computing in the Internet of Things. *Sensors*, 21(11), article number 3800. doi: 10.3390/s21113800.

[15] Liu, B., Buyya, R., & Nadjaran Toosi, A. (2018). A fuzzy-based auto-scaler for web applications in cloud computing environments. In *Service-oriented computing. ICSOC 2018. Lecture notes in computer science* (pp. 797-811). Cham: Springer. doi: 10.1007/978-3-030-03596-9_57.

[16] Llorens-Carrodeguas, A., Leyva-Pupo, I., Cervelló-Pastor, C., Piñeiro, L., & Siddiqui, S. (2021). An SDN-based solution for horizontal auto-scaling and load balancing of transparent VNF clusters. *Sensors*, 21(24), article number 8283. doi: 10.3390/s21248283.

[17] Moura, B.M.P., Schneider, G.B., Yamin, A.C., & Santos, H.S. (2022). Interval-valued fuzzy logic approach for overloaded hosts in consolidation of virtual machines in cloud computing. *Fuzzy Sets and Systems*, 446, 144-166. doi: 10.1016/j.fss.2021.03.001.

[18] Petrakis, E.G.M., Skevakis, V., Eliades, P., Aznavouridis, A., & Tsakos, K. (2024). ModSoftHP: Fuzzy microservices placement in Kubernetes. *Electronics*, 13(1), article number 65. doi: 10.3390/electronics13010065.

[19] Qiu, H., Mao, W., Wang, C., Franke, H., Youssef, A., Kalbarczyk, Z.T., Başar, T., & Iyer, R.K. (2023). AWARE: Automate workload autoscaling with reinforcement learning in production cloud systems. In *Proceedings of the 2023 USENIX annual technical conference* (pp. 387-402). Boston: USENIX.
[20] Sarma, B., Kumar, R., & Tuithung, T. (2021). Optimised fuzzy clustering-based resource scheduling and dynamic load balancing algorithm for fog computing environment. *International Journal of Computational Science and Engineering*, 24(4), 343-353. doi: 10.1504/IJCSE.2021.117015.
[21] Seraj, Y., Fadaei, S., Safaei, B., Javadi, A., Hosseini Monazzah, A.M., & Hemmatyar, A.M.A. (2024). LIMO: Load-balanced offloading with MAPE and particle swarm optimization in mobile fog networks. *ArXiv*. doi: 10.48550/arXiv.2408.14218.
[22] Silva, S.N., Goldbarg, M.A.S.d.S., da Silva, L.M.D., & Fernandes, M.A.C. (2024). Application of fuzzy logic for horizontal scaling in Kubernetes environments within the context of edge computing. *Future Internet*, 16(9), article number 316. doi: 10.3390/fi16090316.
[23] Vashistha, D., Mehta, D., Kumhar, M., Bhatia, J., & Alkhayyat, A. (2025). Deep learning based load balancing in cloud computing: A survey. *Procedia Computer Science*, 259, 1963-1972. doi: 10.1016/j.procs.2025.04.152.
[24] Wu, Y.-J., Chen, M.-C., Hwang, W.-S., & Cheng, M.-H. (2024). Dynamic routing using fuzzy logic for URLLC in 5G networks based on software-defined networking. *Electronics*, 13(18), article number 3694. doi: 10.3390/electronics13183694.
[25] Yi, B., Wang, X., Li, K., Das, S.K., & Huang, M. (2018). A comprehensive survey of network function virtualization. *Computer Networks*, 133, 212-262. doi: 10.1016/j.comnet.2018.01.021.
[26] Zhang, H., & Huang, S.-L. (2025). Improved fuzzy C-means clustering algorithm based on fuzzy particle swarm optimisation for solving data clustering problems. *Mathematics and Computers in Simulation*, 233, 311-329. doi: 10.1016/j.matcom.2025.02.012.
[27] Zhu, S., Zhao, Y., & Yue, S. (2024). Double-constraint fuzzy clustering algorithm. *Applied Sciences*, 14(4), article number 1649. doi: 10.3390/app14041649.

Метод балансування навантаження мережі на основі нечіткої кластеризації

Дмитро Кульчицький
Магістр
Київський національний університет імені Тараса Шевченка
01601, вул. Володимирська, 64/13, м. Київ, Україна
https://orcid.org/0009-0009-4060-315X
Євген Жуков
Ад'юнкт
Національний університет оборони України.
03049, просп. Повітряних Сил, 28, м. Київ, Україна
https://orcid.org/0009-0002-1251-946X

Анотація. Стрімке зростання інтерактивного відео, платформ доповненої та віртуальної реальності й Інтернету речей різко підвищує вимоги до гнучкого розподілу мережевого навантаження. Класичні статичні алгоритми балансування не встигають реагувати на швидку зміну трафікових профілів, що спричиняє перевантаження окремих вузлів. Метою роботи було створити та перевірити ефективність адаптивного алгоритму балансування, заснованого на нечіткому групуванні станів серверів. Методологічно використано нечітку кластеризацію Fuzzy C-Means із формуванням нечітких правил, імітаційне моделювання мережі зі 100 серверами, статистичне порівняння стратегій за метриками завантаженості, часу відповіді та використання ресурсів, аналіз коефіцієнта варіації трафіку й лінійної залежності «кількість кластерів – стабільність», а також чутливий аналіз до похибок вимірювання (≤ 5 %) Спроєктовано модульну систему з підсистемами моніторингу, нечіткої кластеризації методом Fuzzy C-Means і рушія прийняття рішень. Проведено імітаційне моделювання мережі зі 100 серверами та часово змінним навантаженням; статистично порівняно три стратегії: кругове опитування, двопороговий автоскейлінг і розроблений нечітко кластерний підхід. Встановлено, що запропонований алгоритм зменшує середнє відхилення завантаженості вузлів у 1,5 раза порівняно з двопороговим автоскейлінгом. У пікові періоди час відповіді системи скоротився з 180 до 145 мс, а середнє використання ресурсів зросло з 68 до 85 %. Аналіз коефіцієнта варіації трафіку показав, що за значень понад 0,4 новий метод утримує навантаження у межах ±10 % для 92 % спостережень, тоді як кругове опитування виходить за межі у 37 % випадків. Виявлено лінійну залежність між кількістю кластерів і стабільністю розподілу; оптимальним стало значення чотири кластери. Нечіткі правила усувають різкі коливання трафіку після сплесків попиту та забезпечують плавне перенаправлення потоків. Аналіз чутливості довів, що похибка вимірювання метрик до 5 % не впливає на коректність рішень. Побудована архітектура демонструє стабільну роботу при неспівмірному зростанні трафіку на окремих серверах. Порівняння з традиційними методами підтвердило перевагу запропонованого підходу за всіма оціненими показниками продуктивності й стійкості. Розроблену систему можуть упроваджувати оператори дата центрів і хмарних платформ для підвищення надійності послуг із непередбачуваним навантаженням

Ключові слова: комп'ютерні мережі; нормалізація завантаженості серверів; управління змінним трафіком; Fuzzy C-Means; адаптивні алгоритми; штучний інтелект; мережеві вузли

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.03
DOI: 10.62660/bcstu/3.2025.80

Adaptive similarity assessment metric for intelligent failure diagnostics in ship power plants

Vladimir Vychuzhanin*
Doctor of Engineering Sciences, Professor
Odesa Polytechnic National University
65044, 1 Shevchenko Ave., Odesa, Ukraine
https://orcid.org/0000-0002-6302-1832

Alexey Vychuzhanin
PhD, Assistant
Odesa Polytechnic National University
65044, 1 Shevchenko Ave., Odesa, Ukraine
https://orcid.org/0000-0001-8779-2503

Abstract. Prompt and accurate diagnosis of failures in ship power plants (SPPs) is essential for ensuring maritime safety, minimising operational risks, and optimising maintenance strategies. With increasing system complexity, heterogeneous data sources, and limited historical failure records, conventional diagnostic methods often prove insufficient, especially in scenarios involving rare or ambiguous faults. The purpose of this study was to develop an interpretable, adaptive, and probabilistically grounded methodology for assessing similarity between failure cases within SPPs for use in intelligent decision support systems. The proposed method integrates Euclidean, Jaccard, and logistic similarity metrics with Bayesian inference, temporal degradation modelling, frequency-based weight correction, and contextual smoothing of affected subsystems. The model employs L-BFGS-B optimisation to automatically adjust metric weights according to diagnostic relevance. Numerical experiments based on synthetic case data revealed high classification accuracy: 96% for failures related to cooling system overheating, 84% for bearing degradation cases, and 92% for fuel supply irregularities. Even with a 40% reduction in training data volume, the performance drop did not exceed 7%, indicating strong resilience to data sparsity. The visualisation of decision boundaries demonstrated the model's ability to distinguish overlapping failure classes while preserving semantic interpretability. Weight optimisation results identified "failure type" as the dominant factor, while "risk category" and "affected subsystems" had negligible impact and were excluded. Bayesian aggregation further improved the credibility of diagnostic conclusions by combining local similarity with global statistical priors. The developed methodology can be effectively applied by marine engineers, ship operators, and developers of intelligent diagnostic platforms for fault detection, root cause analysis, and predictive maintenance under conditions of uncertainty and incomplete information. Its modular structure also allows extending it to other complex technical domains beyond SPPs

Keywords: decision support system; Bayesian aggregation; adaptive weight coefficients; case-based failure analysis; probabilistic metric; similarity of technical conditions

INTRODUCTION

In modern complex technical systems (CTS), prompt and accurate failure diagnostics of equipment plays a key role in ensuring uninterrupted operation, safety, and high system performance. The increasing complexity and heterogeneity of CTS have driven the widespread adoption of intelligent diagnostic methods based on

Article's History: Received: 18.04.2025; Revised: 03.08.2025; Accepted: 15.09.2025.

Suggested Citation:
Vychuzhanin, V., & Vychuzhanin, A. (2025). Adaptive similarity assessment metric for intelligent failure diagnostics in ship power plants. *Bulletin of Cherkasy State Technological University*, 30(3), 80-92. doi: 10.62660/bcstu/3.2025.80.

*Corresponding author

reusing accumulated experience, among which the case-based reasoning (CBR) approach stands out. Failure diagnosis based on CBR involves comparing the current situation with previously recorded failure cases, where the effectiveness of the similarity assessment between cases is fundamental. Determining similarity between cases is the core task of CBR systems, as the quality of this assessment directly influences the accuracy, relevance, and promptness of diagnostic conclusions. Many approaches have been proposed to improve similarity assessment accuracy – from classical metric functions to hybrid machine learning methods and logical models. However, despite the progress, existing solutions often face a series of limitations: poor scalability, insufficient adaptability to heterogeneous data, and limited interpretability of results.

Adaptive fuzzy logic systems developed by L. Dubchak *et al*. (2024) demonstrated high adaptability and noise resilience in fault detection tasks. However, industrial-scale application is still limited due to the need for further validation on multidimensional heterogeneous datasets. According to S. Ren *et al*. (2020), in the field of low-carbon product design, effective similarity determination between cases plays a critical role. The researchers developed a model that considers both similarity and adaptability for knowledge reuse and design adaptation. B.M. Mathisen *et al*. (2020) argued that similarity metric learning based on data can automate the process of building similarity functions in case-based reasoning systems. The researchers proposed two novel approaches: using a pre-trained classifier as a foundation for the similarity function and a fully data-driven method that minimises training time. Both approaches showed high efficiency on 14 different datasets. Z. Chen *et al*. (2021) demonstrated that graph neural networks (GNN) can reveal hidden structural similarities in cyber-physical systems, but they require complex preprocessing and often suffer from a lack of decision-making transparency. N. Lin *et al*. (2023) proposed an enhancement of the CBR approach through the application of aggregation operators, such as the Choquet integral, which enabled accounting for nonlinear interdependencies between diagnostic features. Despite improving similarity assessment accuracy, this approach complicates the formation of expert preferences. N. Neykov & S. Stefanova (2023) demonstrated the applicability of rule-based heuristics in CBR systems, highlighting their flexibility and explainability, but also noted challenges when working with large volumes of cases and the need for manual attribute weight tuning. According to N. Lin *et al*. (2023), interpretability plays a crucial role in case similarity matching systems in the legal domain. The researchers proposed an integrated framework consisting of four modules: key sentence identification, case matching, sentence alignment, and conflict resolution. This approach ensures transparency and explainability of the matching process. Most existing models do not account for probabilistic dependencies between failures, differences in operating conditions, and the dynamic evolution of the technical state. For example, one failure may increase the probability of subsequent faults; operating modes may accelerate equipment degradation; component conditions change over time, influencing failure forecasting.

To enable accurate identification of typical equipment failures and their root causes within systems such as SPP, it is necessary to develop a unified and adaptive methodology for assessing the similarity between cases. At its core, the methodology should feature a similarity metric that ensures relevant comparison of current cases with archived ones and identification of the most similar analogues. In this context, similar cases refer to those with comparable operating conditions, identical damage characteristics, close parameter values, and matching probabilistic characteristics. For example, two main engine failures can be considered similar if they occurred under comparable operating hours, were accompanied by similar changes in temperature and pressure parameters (including oil temperature and fuel pressure), and shared a root cause – for instance, overheating caused by cooling system failure. The methodology being developed should account for both quantitative and qualitative failure features; enable the integration of metric, logical, and machine-learning components; and be applicable to heterogeneous data types (numerical, categorical, set-based, and temporal). Other requirements include interpretability for technical experts and adaptability to changing operating conditions.

The proposed structured approach covered industry-specific characteristics, applied parameter normalisation, and implemented weight coefficient optimisation to improve diagnostic reliability. Particular attention was given to considering operational and technical characteristics (load, operating time, type of damage), causal relationships between components, statistical parameters (failure frequency, prior probabilities), and the predicted dynamics of equipment conditions. Altogether, this enabled the construction of an adaptive case-matching system capable of effectively identifying similar failures under real-world operating conditions.

The purpose of this study was to develop an interpretable, adaptive, and statistically grounded methodology for assessing similarity between failure cases in ship power plant systems, based on the integration of metric, probabilistic, and logic-based components. The methodology was intended to improve matching accuracy, robustness against heterogeneous data, and adaptability to changing operational conditions, with subsequent integration into intelligent CBR-based diagnostic systems.

MATERIALS AND METHODS

This study examined and compared various approaches to assessing the degree of similarity between equipment failure cases (precedents) within SPP. The focus was on the analysis of similarity metrics used within

the CBR methodology, including conventional, probabilistic, and hybrid approaches. The analysis covered the following categories of methods: distance metrics, such as Euclidean, Manhattan, and cosine, which were applied to numerical and categorical parameters; set-based metrics, including the Jaccard index, which were used to compare sets of affected subsystems and risk categories; probabilistic approaches, where the similarity metric was adjusted based on the empirical probability of joint feature occurrence or by using Bayesian weights; aggregated models, where the final similarity metric represented a weighted sum of partial metrics with weights optimised numerically (using the L-BFGS-B method); and hybrid CBR structures that combined rules, expert assessments, and supervised learning.

To evaluate the effectiveness and identify the limitations of the listed methods, a comparative experimental series was conducted. Within this series, a synthetic training sample of precedents was created, covering typical equipment failures of SPP, involved components, and occurrence probabilities. Baseline methods with fixed weights were compared to adaptive models, where weights were adjusted based on comparison with expert similarity assessments. The effectiveness of the models was evaluated according to the following criteria: classification accuracy, including the use of confusion matrices; correlation with expert similarity assessments; and sensitivity of metrics to variations in input parameters.

The parameter weights were optimised using the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with bound constraints, where the loss function was defined as the mean squared error between the predicted and expert-assessed similarity values. For this, the implementation from the scipy.optimize library was used, ensuring computational efficiency and reproducibility. Numerical experiments with parameter variation and visualisation tools that demonstrated the contribution of individual components to the final similarity measure were applied for model validation. The presented methodology helped to identify key limitations of conventional approaches and demonstrated the advantages of adaptive, probabilistically informed models in failure diagnostics.

To search for similar failures in the precedent database, the k-Nearest Neighbours (k-NN) method was used, as described by M. Zuber & R. Sirdey (2021). This approach helped to identify the most similar cases based on a predefined similarity metric. The choice of this method was driven by the following advantages: flexibility – k-NN handled heterogeneous data types (numerical, categorical, probabilistic parameters); interpretability – unlike neural network models, it was possible to explain why two failures were considered similar; adaptability – the method allowed adjusting parameter weights depending on operating conditions; efficiency with limited data – it did not require a large training set, which was critical for the diagnostics of rare failures. The similarity between two precedents was calculated based on a weighted sum of partial similarities across key parameters, as presented in Table 1.

Table 1. Similarities between two precedents based on key parameters

Parameter	Description	Similarity Measurement Method
Failure Type	Nature of damage, degradation mechanism	Cosine similarity
Failure Causes	External and internal factors, preceding events	Jaccard similarity
Failure Probability	Estimated probability of failure	Euclidean distance
Failure Consequences	Impact on operability, cascading effects	Bayesian networks
Affected Components	Failed nodes and subsystems	Jaccard similarity

Source: created by the authors of this study

Depending on the type of data, the following metrics were used: euclidean distance – for numerical parameters (e.g., failure probability, remaining useful life); cosine similarity – for textual and vector representations of failure categories; jaccard similarity – for categorical data (e.g., lists of affected subsystems); Markov models – for predicting failure evolution over time and adjusting parameter significance; Bayesian networks – for accounting for probabilistic dependencies between failures.

RESULTS AND DISCUSSION

To build a formalised model for determining similarity between precedents, information on typical failures of SPP equipment and corresponding diagnostic features was structured. Table 2 presents the classification of failures according to key parameters: temperature; vibration level; pressure; typical manifestations. This structure allows highlighting diagnostically significant features, which are further formalised into precedent parameters. Additionally, Figure 1 presents a classification error matrix that reflects characteristic cases of misclassification of observed features into failure types, demonstrating the limitations of conventional matching methods and emphasising the need for a more precise similarity evaluation model.

Figure 1 presents the classification error matrix, reflecting the performance of the CBR system when classifying new precedents by failure types based on the applied similarity assessment methodology.

Table 2. Classification of typical failures and diagnostic features

Failure Type	Temperature	Vibration	Pressure	Typical Symptoms
Failure A	> 90°C	2.0-3.5 mm/s	> 10 bar	Often associated with cooling system overheating
Failure B	75-85°C	1.5-2.5 mm/s	6-9 bar	Related to gradual bearing wear
Failure C	> 85°C	2.5-4.0 mm/s	9-11 bar	Observed with unstable fuel supply

Source: created by the authors of this study

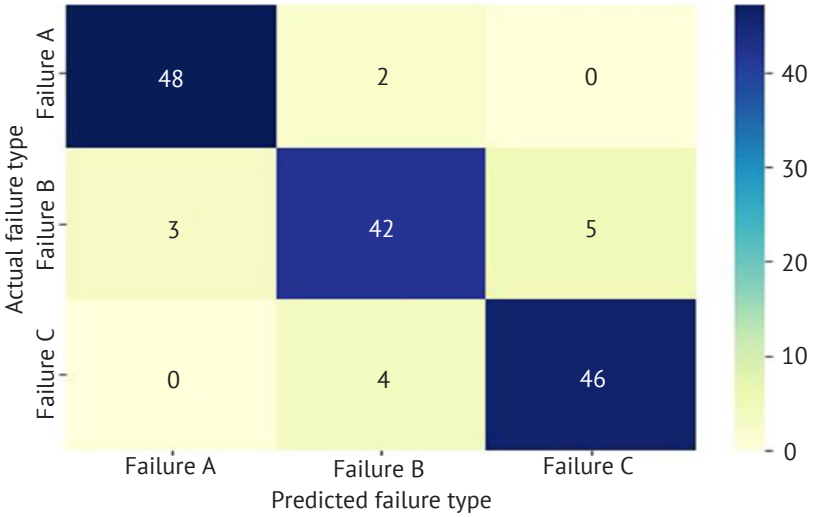


Figure 1. Classification error matrix by failure types

Source: created by the authors of this study

The factual and predicted failure types (A, B, C) are indicated along the axes (Fig. 1). The values on the diagonal of the matrix correspond to correctly classified cases: 48 instances of type A were correctly identified as A; 42 instances of type B were classified correctly; and 46 instances of type C were recognised accurately. Classification errors were observed in the following forms: 3 cases of type B were mistakenly assigned to class A, and 5 to class C; 4 cases of type C were classified as B. Type A demonstrated the greatest classification stability (only 2 errors out of 50 observations, accuracy – 96%). This reflects high overall classification accuracy, with the share of correct predictions exceeding 90% for all failure types. A notably increased probability of cross-classification between types B and C may be explained by partial overlap of features (e.g., similar vibration and pressure values under different failure scenarios); insufficient discrimination capacity of the current similarity metric in the feature space of these types or a limited training sample size for B and C cases. Such insights enable targeted model enhancement – for instance, by refining parameter weights or introducing other features that enable clearer separation of failure types with similar manifestations.

The developed similarity model for comparing a current case A_i and a precedent B_i is represented as follows:

$$S(A_i, B_i) = \alpha_{s.i} \cdot d_{types.i}(A_i, B_i) + \beta_{s.i} \cdot d_{probability.i}(A_i, B_i) + \gamma_{s.i}, d_{components.i}(A_i, B_i), \quad (1)$$

where $d_{types.i}(A_i, B_i)$ is the similarity measure for failure types; $d_{probability.i}(A_i, B_i)$ is the difference in failure probabilities; $d_{components.i}(A_i, B_i)$ is the similarity estimation for the affected components; $\alpha_{s.i}$, $\beta_{s.i}$, $\gamma_{s.i}$ are the weighting coefficients that determine the significance of different aspects when comparing precedents

The similarity components are calculated as follows:

$$d_{types.i}(A_i, B_i) = \begin{cases} 1, if\ types\ are\ identical \\ 0.5, if\ in\ the\ same\ category; \\ 0, if\ types\ differ \end{cases} \quad (2)$$

$$d_{probability.i}(A_i, B_i) = 1 - |\mathrm{P}(A_i) - \mathrm{P}(B_i)|; \quad (3)$$

$$d_{components.i}(A_i, B_i) = \frac{|\mathrm{C}(A_i) \cap \mathrm{C}(B_i)|}{|\mathrm{C}(A_i) \cup \mathrm{C}(B_i)|}, \quad (4)$$

where $P(A_i)$, (B_i) are the failure probabilities for the respective components; $C(A_i)$, $C(B_i)$ are the sets of components involved in the failures. This formulation enables a quantitative accounting of subsystem overlap, which directly affects the assessment of case similarity.

For normalisation of the final similarity metric, the following expression is applied:

$$D_{total}(A_i, B_i) = = \frac{\alpha_{s.i} \cdot d_{types.i}(A_i, B_i) + \beta_{s.i} \cdot d_{probability.i}(A_i, B_i) + \gamma_{s.i} \cdot d_{components.i}(A_i, B_i)}{\alpha_{s.i} + \beta_{s.i} + \gamma_{s.i}}. \quad (5)$$

Such normalisation ensures that the resulting similarity score stays within the range [0, 1], enabling

consistent comparison of different case pairs and facilitating further interpretation within the diagnostic decision-making process. The coefficients $\alpha_{s.i}$, $\beta_{s.i}$ and $\gamma_{s.i}$ are optimised using numerical methods aimed at minimising a loss function that reflects the classification error based on known labels (e.g., the mean squared error between the predicted similarity value and the expert-assessed value). This allows the model to be adapted to the specifics of a particular domain and the structure of the case database. To improve the accuracy, interpretability, and adaptability of the similarity assessment model between cases, the following enhancements have been implemented:

Weight correction based on failure frequency

This extension allows the model to factor in the rarity of specific failure types in the dataset, automatically increasing the importance of rarely occurring but potentially critical scenarios. This is particularly relevant for the diagnostics of low-probability but high-risk conditions, which standard methods may overlook due to their low occurrence rate.

The weight correction formula is defined as:

$$\tilde{\alpha}_{s.i} = \alpha_{s.i} \cdot \log\left(1 + \frac{1}{f_{type}(A_i)}\right), \quad (6)$$

where $f_{type}(A_i)$ is the relative frequency of occurrence of failure type A_i in the case database.

This adjustment ensures automatic balancing of the model: the rarer a particular failure type, the greater its weight when calculating the final similarity measure. This enables the model to better handle the diagnostics of infrequent but significant cases, preventing bias toward frequently occurring but less critical failures. Such a correction can further be extended to other parameters, including probabilistic and structural characteristics, which would additionally enhance the flexibility and reliability of the case-based reasoning system.

Temporal degradation factor

This extension accounts for the influence of the temporal aspect (e.g., the difference in operating time until failure) when assessing the similarity between cases. Incorporating this factor allows the model to prioritise cases that are closer in terms of operational time, making the reasoning process more context-aware. The temporal degradation factor is calculated as follows:

$$d_{degradation}(A_i, B_i) = exp(-\lambda \cdot |TA_i - TB_i|), \quad (7)$$

where TA_i, TB_i are the operating times until failure for cases A_i and B_i; λ is the degradation coefficient, which controls the sensitivity of the model to temporal differences.

The adjusted similarity score is defined as follows:

$$S'(A_i, B_i) = S(A_i, B_i) \cdot d_{degradation}(A_i, B_i). \quad (8)$$

The introduction of this multiplicative temporal context reflects the idea that two cases with identical features but differing significantly in failure time are diagnostically less similar than cases with both matching features and close failure times. Figure 2 illustrates how the temporal degradation coefficient varies depending on the difference in operational time between two cases. The figure presents three scenarios corresponding to different sensitivity levels of the model to temporal discrepancies, controlled by the value of the λ parameter.

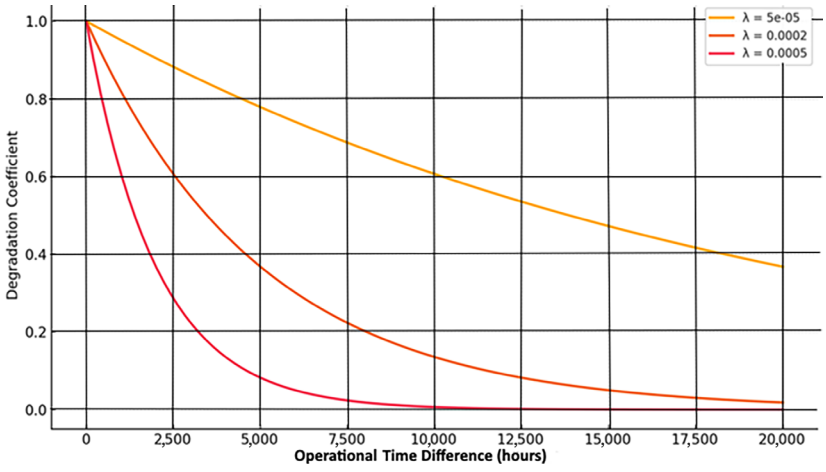


Figure 2. Change of the temporal degradation coefficient depending on the difference in operating time between two cases

Source: created by the authors of this study

The curve with the smallest λ (0.00005) shows a slow decrease of the coefficient, which reflects a model with stable memory – it "trusts" even older cases. Conversely, at a large λ (0.0005), the model quickly depreciates old cases, focusing on the most recent data. Thus, the choice of λ directly affects the behaviour of the model: a low λ implies high stability, which is useful in the presence of stable patterns; a high λ ensures adaptability to new conditions, which is necessary in the case of dynamically changing technical states of the SPP system. This

emphasises the significance of calibrating the temporal component when designing CBR systems, especially in the context of variable load and operating modes.

Contextual smoothing of components supplements the comparison of SPP system components with information about the proximity between components (e.g., if two components are technically related). This reduces information loss in cases of partial mismatch.

$$d_{comp}^{adj} = \frac{|C(A_i) \cap C(B_i)| + \delta \cdot |Ad_j(A_i, B_i)|}{|C(A_i) \cup C(B_i)| + \delta \cdot N}, \quad (8)$$

where $|Ad_j(A_i, B_j)|$ is the number of adjacent, functionally related subsystems; δ is the smoothing coefficient; N is the total number of possible components.

Nonlinear activation (e.g., tanh) allows considering the saturation effect in cases of strong similarity and helps to soften sharp boundaries. This is used for better interpretation of the final similarity in extreme cases:

$$S(A_i, B_i) = \tanh(\alpha_{s.i} \cdot d_{types.i}(A_i, B_i) + \beta_{s.i} \cdot d_{probability.i}(A_i, B_i) + \gamma_{s.i} \cdot d_{components.i}(A_i, B_i)). \quad (6)$$

The proposed formalised model for assessing similarity between cases (1) is integrated into the classical CBR architecture, which includes four key stages: retrieval, adaptation, application, and update. At the retrieval stage, the model is used to search for the most relevant analogues in the case base.

Stage 1. Calculation of similarity between cases

For a new diagnostic case A_i, represented by a feature structure (failure type, probability, subsystems, etc.), the similarity measure is calculated for all cases B_j from the knowledge base according to formula (1). If additional modules (temporal degradation, probabilistic correction, etc.) are activated, the final value is modified accordingly:

$$S'_{norm}(A_i, B_i) = \frac{S(A_i, B_i) \cdot d_{degradation}(A_i, B_i) \cdot P_i(A_i, B_i)}{\alpha_{s.i} + \beta_{s.i} + \gamma_{s.i}}, \quad (9)$$

where $Pi(Ai, Bi) \in |0,1$ is the probability of co-occurrence of the values A_i and B_i in the cases, extracted from empirical data or an expert model

Stage 2. Nearest neighbour search (k-NN)

Based on the obtained values $S(A_i, B_i)$, the k nearest cases with the highest similarity values are selected. After constructing and formalising the similarity metric between cases across the set of parameters, the presented model is integrated into the k-Nearest Neighbours (k-NN) search algorithm. Using k-NN not only enables the application of the similarity metric but also implements a decision-making process based on it.

Advantages of the approach: local analysis: decisions are made based on similar cases, which is especially significant considering the high variability of operating modes of marine power plants; flexibility: the method can be adapted for specific diagnostic goals (e.g., predicting time to failure or classifying the failure type); scalability: adding new cases does not require restructuring the model, only a local recalculation of similarity. For the diagnostics of a new case (the observed equipment condition): the global similarity measure $S(A_i, B_i)$ is calculated between the new case and each case in the database according to formula (1); the k cases with the highest similarity (equivalently – with the minimum distance according to the metric) are selected. A diagnostic conclusion is then formed based on the k nearest cases. Possible approaches include majority voting – the most frequently occurring failure type among the nearest neighbours; weighted voting – considering the similarity level of each neighbour; aggregated parameter values – e.g., averaging the time to failure, temperatures, pressures, etc. The application of k-NN requires prior normalisation of the parameters involved in the calculation of individual similarity measures to avoid domination by parameters with larger measurement scales. Additionally, the choice of the parameter k should factor in the density of the case base (with low density, a smaller k is preferable); the variability and noise level in the data (for high noise, it is better to average over a larger number of neighbours); the type of task being solved (classification, prediction, anomaly detection, etc.). Figure 3 illustrates the nearest neighbour method (k-NN) in the parameter space that defines the similarity between cases.

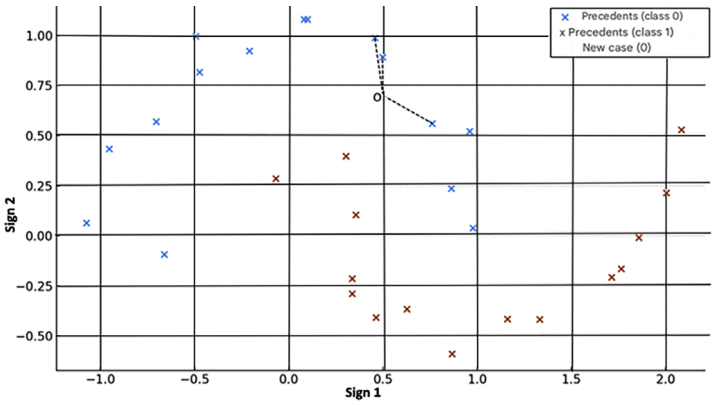


Figure 3. Application of the k-NN method in the parameter space defining the similarity between cases
Source: created by the authors of this study

Figure 3 presents an example of classifying a new failure case with coordinates (0.5; 0.7) in the feature space using the k-nearest neighbours method ($k = 3$). The dashed lines visualise the connection of the analysed object with the three most similar cases, selected according to the minimum distance metric in a pre-normalised feature space with consideration of their weighting coefficients. The position of the new case in the area dominated by cases of the first class allows the model to classify it as a failure of the first type based on the neighbour voting strategy. This example demonstrates the key advantages of the k-NN method for diagnosing complex technical systems: the ability to work with nonlinearly separable data, robustness to noise through aggregation of information from several neighbours, as well as clear interpretability of the results.

Special attention must be paid to the correct tuning of the method's critical parameters – the optimal k value, proper normalisation of features, and the selection of an appropriate similarity metric that considers the scale and informativeness of various parameters. The presented approach ensures the consideration of the local distribution characteristics of the data and increases the robustness of the results to variability in input parameters. Its ability to work with incomplete and noisy data, while combining visual representation with probabilistic justification of decisions, makes the method especially valuable for diagnosing SPP, where one typically faces limited historical data, complex nonlinear dependencies, and high demands for interpretability of the obtained results.

Stage 3. Aggregation of the diagnostic decision

The extracted k-nearest neighbours are analysed: by the most frequently occurring failure type (majority voting); by the average probability of failure; by contextually weighted characteristics. The final diagnostic conclusion is formed based on the coordinated information from the most similar historical cases. Once the k most similar cases $\{B_1,\ldots,B_i\}$ have been determined for the new case A_i, their diagnostic characteristics are aggregated. The most typical strategies include majority voting for failure type (the failure type that occurs more frequently among the nearest cases is selected):

$$T^* = arg\ max_T \sum_{i=1}^{k} 1T_i = T. \quad (10)$$

To determine the most probable failure type T^* among the nearest cases, a majority voting strategy is applied. The indicator function $1_{T_i = T}$ equals 1 if the failure type T_i in the i^{th} precedent matches type T, and 0 otherwise. By summing these indicators across all k nearest neighbours, the method identifies the most frequently occurring failure type. The type T with the highest number of "votes" is selected as the final diagnosis.

Weighted average failure probability:

$$\hat{p}(A_i) = \frac{1}{\sum_{i=1}^{k} S(A_i,B_i)} \cdot \sum_{i=1}^{k} S(A_i,B_i) \cdot p(B_i), \quad (11)$$

where $p(B_i)$ is the known failure probability for the corresponding precedent.

To estimate the failure probability for the current case A_i, a weighted average of known failure probabilities $p(B_i)$ from the k nearest precedents is used. Each precedent's contribution is weighted by its similarity score $S(A_i, B_i)$. The denominator normalises the sum, ensuring the resulting probability stays within the [0, 1] range. This approach gives more influence to precedents that are more similar to the current case.

Merging of affected components

If failure localisation is required, it is possible to form an aggregated set of the most probable components affected in similar cases: $C_{agg}(A_i) = \bigcup_{i=1}^{k} C(B_i)$. When the objective is failure localisation, an aggregated set of potentially affected components is formed by taking the union of component sets $C(B_i)$ from the k most similar precedents. This provides a complete list of all subsystems that were involved in similar past failures and may be relevant for the current case A_i.

The final aggregated set of components (subsystems) that are considered most likely involved in the failure for the current case A_i, accounting for weights (similarity):

$$C_{agg}^{score}(A_i) = \{c: \sum_{i=1}^{k} S(A_i,B_i) \cdot 1_{c \epsilon C(B_i)} > \theta\}, \quad (12)$$

where c is a specific subsystem of the SPP (e.g., "fuel pump," "cooling circuit," etc.) considered as potentially failed; $\sum_{i=1}^{k} S(A_i, B_i) \cdot 1_{c \epsilon C(B_i)}$ is the weighted sum over k nearest precedents; $S(A_i, B_i)$ is the similarity between the current case and precedent B_i; $1_{c \epsilon C(B_i)}$ is the indicator function: equals 1 if component is present in the list of failed subsystems of precedent B_i, and 0 if not; θ is thesignificance threshold, which determines how frequently and with what weight component c must occur to be considered significant.

To improve failure localisation accuracy, this formula performs a weighted aggregation of components based on their presence in similar cases. A component c is included in the final set $C_{agg}^{score}(A_i)$ if its weighted occurrence frequency across the k nearest neighbours exceeds a given threshold θ. This approach helps filter out random or weakly relevant components, retaining only those that consistently and significantly appear in similar cases.

Although the primary goal of the proposed model was to construct a similarity metric between cases, the implementation of diagnostic inference in a CBR system requires a justified decision-making method. When several similar cases are available, it becomes necessary to aggregate their characteristics. In such conditions, it is advisable to apply a probabilistic approach – specifically, Bayesian aggregation – for interpreting results and improving diagnostic accuracy.

This model extension logically followed from the task of result interpretation after computing the similarity. Additionally, the diagnostic decision can be

justified in terms of Bayesian inference, where the failure type T was treated as a hypothesis, and the retrieved precedents as observations:

$$P(T \mid A_i)_k \mid) \propto P(T) \cdot \prod_{i=1}^{k} P(B_i \mid T)^{\omega_i}, \quad (13)$$

where $P(T)$ is the prior probability of failure type T (according to the statistics of the precedent database); $P(B_i \mid T)$ is the conditional probability of observing precedent B_i given failure type T; $\omega_i \propto S(A_i, B_i)$ is a weight reflecting the confidence level in precedent B_i based on its similarity to A_i.

If it is assumed that the probability of observing a specific precedent B_i depends on its degree of similarity to A_i, then:

$$P(B_i \mid \mathrm{T}) \approx \mathrm{S}(A_i, B_i) \cdot 1(T_i = T), \quad (14)$$

which leads to the following posterior estimation:

$$P(T \mid A_i)\, k \mid) \propto P(T) \cdot \sum_{i=1}^{k} \mathrm{S}(A_i, B_i) \cdot 1_{T_i = T}. \quad (15)$$

In practice, this means that the failure type with the highest posterior probability is accepted as the most probable diagnosis for the current case A_i. This approach is particularly useful under conditions of incomplete data, where precedents may be heterogeneous or lack a complete set of features, and the Bayesian model allows systematically incorporating the degree of confidence. Figure 4 presents a visualisation of the posterior inference process for determining the probability of a new precedent (observed failure) belonging to one of the typical failure classes (e.g., A, B, C) based on the Bayesian model.

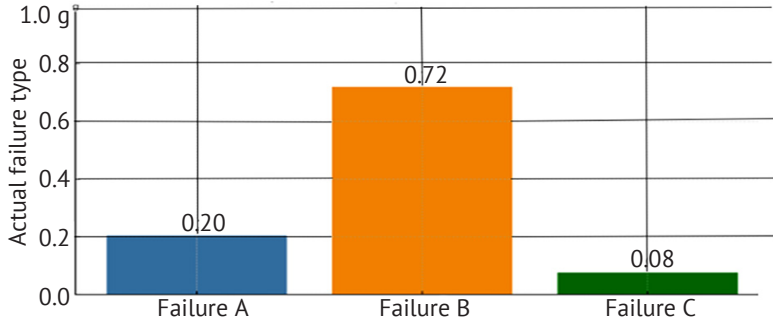


Figure 4. Visualisation of the posterior inference process for determining the probability of a new precedent's class affiliation

Source: created by the authors of this study

Bayesian aggregation was applied here as an interpretable decision-making mechanism, enabling not only the identification of the most probable failure type but also the quantitative consideration of uncertainty and the probabilistic distribution of hypotheses. The diagram illustrates that, for the analysed case, the greatest posterior probability corresponds to failure type B. This reflects both the frequency dominance of similar cases of type B among the nearest neighbours (resulting from the k-NN search in the similarity space) and the high prior probability of this failure type within the overall diagnostic statistics of the precedent database. The Bayesian aggregation methodology enables the weighted integration of two key sources of information: global knowledge about the distribution of failures (prior probabilities); local similarity of the current case to recorded precedents.

This model was particularly effective when working with limited training datasets, as well as in scenarios with diverse failure causes, where the same type of malfunction can result from various degradation pathways. The use of probabilistic inference: enhances interpretability – each step of the decision-making process can be justified through the probabilistic model; reduces sensitivity to data noise – the unequal contribution of neighbours is regulated not only by similarity but also by their probabilistic weight; allows for distributed uncertainty between multiple failure types – especially in the presence of overlapping features. Thus, the visualisation confirms the validity of using Bayesian aggregation as an extension to the k-NN method, providing more accurate, flexible, and robust classification of precedents in the intelligent diagnosis of SPPs.

Stage 4. Updating the case database

After the diagnosis has been verified, the new case – including the original feature set and diagnostic results – was added to the case database for further enrichment and continuous model training. As the database of precedents grows, the system became increasingly capable of accurately matching new cases to previously observed ones, progressively adapting the model to real-world operating conditions. During the classification and decision-making stages, the mechanism of comparison with the most similar precedents played a crucial role.

Thus, the proposed similarity assessment model for failure precedents in SPPs represented an integrated system that combines numerical, categorical, and multi-valued parameters with adaptive weights and specialised similarity metrics. The stepwise extension of the base formula – through the inclusion of failure frequency, temporal factors, functional interconnections of

subsystems, and nonlinear transformations – ensured not only high accuracy but also the robustness of the model against data heterogeneity. The additional application of probabilistic aggregation for determining the final diagnostic conclusion demonstrated the potential for integrating this model into broader intelligent CBR systems. This integration opens new opportunities for enhancing diagnostic reliability and decision interpretability, especially under conditions of limited training data or high variability in failure mechanisms.

The optimisation of parameter weights enabled the model to consider the relative significance of each feature for diagnostic purposes. The coefficients„ were optimised using a bounded limited-memory quasi-Newton optimisation method (L-BFGS-B algorithm), which minimised the following loss function:

$$J(\alpha_{s.i}, \beta_{s.i}, \gamma_{s.i}) = \sum_i^M (S_{norm}(A, B_i) - y_i)^2, \quad (16)$$

where y_i is a binary label assigned in the training dataset of M known pairs, indicating whether the precedents are considered similar (1) or dissimilar (0). When formulating the optimisation problem, there is a need to assume the training dataset consists of N pairs of precedents (i, j), for which the reference similarity degree $S_{i,j}^{ref}$ has been assigned by an expert. For each pair, the model computes the predicted similarity value $S_{i,j}^{mod}(\omega)$, which depends on the weight *vector* $\omega = (\omega_1, \omega_2, \ldots, \omega_k)$. The objective function for optimisation is defined as follows:

$$L(\omega) = \frac{1}{N} \cdot \sum_{i,j} (S_{i,j}^{mod}(\omega) - S_{i,j}^{ref})^2, \quad (17)$$

where the minimised value $L(\omega)$ represents the mean squared deviation between the model-predicted and expert-defined similarity values. The L-BFGS-B algorithm enabled efficient minimisation of this function under constraints on the weight parameters (e.g., $\omega_i \geq 0$, $\sum \omega_i = 1$). The optimisation procedure was implemented using the L-BFGS-B method from the scipy.optimize library (Virtanen *et al.*, 2020). The optimisation process consisted of several stages: formation of a training dataset containing pairs of precedents with known expert similarity assessments; initialisation of weight values; and evaluation of the optimisation quality using a separate test dataset. Figure 5 presents a diagram comparing the initial and optimised weights.

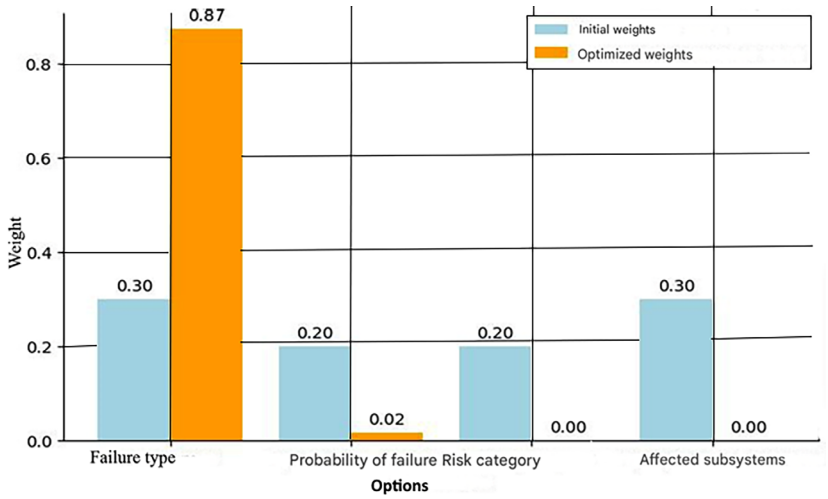


Figure 5. Comparison of initial and optimised weights

Source: created by the authors of this study

The diagram illustrates the comparison between the initial and optimised weights of various parameters used in the similarity assessment of failure precedents. A significant increase in the significance of the "failure type" parameter was observed: initial weight – 0.30; optimised weight – 0.87. This indicates that the parameter has a strong impact on diagnostic accuracy, and as a result, its contribution to the model was substantially increased. A reduction in the influence of the "failure probability" parameter was recorded: initial weight – 0.20; optimised weight – 0.02. This suggests that failure probability has a lower significance when determining precedent similarity, and its weight was nearly nullified.

The "risk category" and "affected subsystems" parameters were excluded from the final model. Their initial weights were 0.20 and 0.30, respectively, but after optimisation, both were reduced to 0.00. This suggests that these parameters have no meaningful influence on the diagnostic outcome and can be safely removed from the model. In summary, the optimisation process highlighted that the "failure type" parameter is the key factor when comparing failure precedents. In contrast, parameters such as the equipment risk category and affected subsystems were found to be insignificant and were eliminated from the model. It is essential to verify whether such a radical redistribution of weights leads to a loss of meaningful diagnostic features. It may also be advisable to explore alternative optimisation strategies that preserve the contribution of all parameters to a certain extent. To assess the position of the proposed

model for evaluating similarity between failure cases within the SPP equipment diagnostic structure, an in-depth analysis of analogous studies was conducted. The comparison focused on key methodological dimensions: the nature of the similarity metric, integration of probabilistic reasoning, presence of adaptive weight mechanisms, robustness to incomplete or noisy data, and applicability across heterogeneous technical domains.

In the analysis of time series sensor data, J. Serrà & J.Ll. Arcos (2014) studied the distance measures such as dynamic time warping (DTW), confirming their robustness against temporal distortions. Nevertheless, such methods perform poorly in semantically saturated contexts and are computationally intensive. In another area, J. Ye (2017) introduced neutrosophic logic with cotangent-based similarity measures to address uncertainty in technical system data, although result interpretation continues to be challenging.

The DeepKAF approach presented by K. Amin *et al.* (2020) effectively merged deep learning and CBR for NLP tasks but lacked flexibility in similarity metric configuration. While this model demonstrated high performance in pattern recognition, it relied on rigid internal structures and did not support metric reconfiguration, thus limiting its applicability in domains requiring feature-specific interpretability. In contrast, the model presented in the present study enables flexible configuration of similarity metrics for numerical, categorical, and set-valued attributes, improving transparency and domain adaptability. D. Leake *et al.* (2021) focused on neural case adaptation to enhance analogical reasoning. However, presented approach lacked support for probabilistic inference and dynamic weight adjustment, which are essential when dealing with sparse or uncertain failure data. The model under discussion addressed this gap by incorporating Bayesian hypothesis aggregation and data-driven optimisation of feature weights, improving reliability and resilience to data imperfections. M. Chen *et al.* (2022) proposed an adaptive metric learning system for aeroengine fault diagnostics. Their findings confirmed the benefits of adaptation in technical diagnostics, yet the proposed architecture did not explicitly incorporate probabilistic modelling or failure frequency weighting. In contrast, the method evaluated in this study links metric behaviour to statistical failure patterns and employs Bayesian reasoning to support decision-making under uncertainty. M.F. Valdez-Ávila *et al.* (2023) developed the CBR-fox model for time-series analysis. While effective for temporal patterns, presented solution did not support integration of diverse static features or account for temporal degradation. The approach discussed in this paper expanded these capabilities by incorporating set-based feature fusion and temporal decay modelling, enhancing long-term diagnostic accuracy. M. Krüger (2023) applied similarity measures to cybercrime profiling. Although effective in its domain, the approach was specialised and did not demonstrate cross-domain flexibility. The presented similarity model was designed to account for a variety of technical systems, including electromechanical and thermal areas, which is necessary for wider application in engineering. H. Xu *et al.* (2023) explored the integration of CBR with ontological knowledge graphs for bridge safety diagnostics. Their study offered semantic depth but required complex, manually curated graphs, limiting scalability. The current model bypassed such dependencies by directly handling generalised structured features, enabling more efficient deployment in real-time scenarios. X. Ye *et al.* (2024) introduced neural k-NN into CBR for enhanced representation learning. Despite improved accuracy, the use of implicit neural weights reduced transparency and limits practical interpretability. In contrast, the present approach employed explicitly defined weights with tunable parameters based on operational significance, maintaining explainability. S. Mustyala & M. Bisi (2025) applied harmony search combined with CBR for software effort estimation. Although adaptive in nature, the model lacked probabilistic mechanisms, which are critical for uncertainty modelling. This limitation was addressed in the current framework through Bayesian aggregation and normalisation based on similarity strength. G. Jimenez-Diaz & B. Díaz-Agudo (2024) developed visualisation tools for understanding similarity structures in CBR. Their study improved user transparency but did not integrate probabilistic interpretation. In the presented model, visualisation was combined with statistical weighting, enhancing both interpretability and reliability. T. Zeng *et al.* (2025) proposed the multi-source feature fusion case-based reasoning (MSFF-CBR) model for diagnostic applications involving multi-modal data. While successful in probabilistic fusion, the model devoted less attention to weight interpretability and frequency-based adaptation. The methodology discussed here strengthens this dimension by introducing weight calibration based on historical failure data and component-level impact.

This comparative review highlighted that while many modern approaches bring valuable innovations such as hybrid neural-CBR models, ontology integration, or time-series analytics only a few effectively combine probabilistic reasoning, adaptive weights, support for heterogeneous data, and interpretability. The presented model achieved such integration through a structured framework based on: configurable partial similarity metrics (Euclidean, Jaccard, logistic); gradient-based optimisation of weights using the L-BFGS-B method; Bayesian aggregation for probabilistic inference; support for structured numerical, categorical, and set-valued data. This methodological composition is particularly well-suited for complex environments such as SPPs, where uncertainty, data sparsity, and multi-source features are the norm. The architecture also enabled modular extensions to additional components or operational parameters without requiring full retraining, which is beneficial for maintenance and scalability. Furthermore, the approach aligned with current trends in intelligent

diagnostics, which emphasised interpretability and adaptability over opaque black-box learning. While conventional systems often rely on fixed similarity metrics or rigid neural models, the model discussed in the present study prioritised traceability, domain adaptability, and evidence-based decision-making. The resulting advantages can be summarised as follows: data comprehensiveness: simultaneous handling of numeric, categorical, and set-valued features; adaptability: numerical optimisation of similarity weights through formal procedures; probabilistic robustness: inference supported by Bayesian aggregation and normalisation; scalability: application across diverse diagnostic domains with minimal reconfiguration. In summary, the proposed model constitutes a substantial advancement in the domain of intelligent similarity-based diagnostics. It responded directly to limitations identified in current literature and offers a reliable, interpretable, and extensible solution suitable for decision support systems operating under conditions of complexity and uncertainty.

CONCLUSIONS

The developed adaptive similarity metric for failure diagnostics in SPPs demonstrated significant potential for enhancing both the accuracy and interpretability of intelligent decision support systems. The key contributions of this study included a comprehensive approach to similarity assessment by integrating heterogeneous data types: numerical; categorical; set-based through a weighted combination of metrics such as Euclidean, Jaccard, and logistic. The model employed automatic optimisation of weight coefficients using the L-BFGS-B method, which ensured adaptability to varying operational conditions. Interpretability and robustness were improved through the introduction of weight correction mechanisms that accounted for failure frequency, temporal degradation (based on operating time), and contextual smoothing across components. These enhancements allowed the model to effectively handle rare yet critical failure scenarios. Additionally, the application of non-linear activation functions (tanh) and Bayesian hypothesis aggregation contributed to more accurate classification results and reduced sensitivity to noisy and incomplete data. Experimental evaluation confirmed the practical effectiveness of the proposed model. The classification accuracy exceeded 90% for all three main types of failures (A, B, and C), with Type A achieving 96% correct identification, Type B – 84%, and Type C – 92%, as shown in the confusion matrix. The optimised model also demonstrated strong robustness under conditions of limited training data. For instance, reducing the training set size by 40% led to only a 5-7% drop in classification performance, confirming the model's data efficiency. The visualisation of feature spaces, impact of parameter weighting, and temporal degradation coefficients illustrated the model's ability to separate nonlinear and overlapping classes while preserving explainability.

Compared to conventional methods with static metrics, the proposed approach successfully combined flexibility through adaptability to heterogeneous data with explainability, via Bayesian reasoning and tunable parameters. Its results were comparable to those achieved by ultramodern hybrid techniques such as graph neural networks, but with lower computational requirements and simpler implementation. The developed methodology was considered suitable for integration into CBR-based diagnostic systems aimed at automating the identification of failure types, forecasting the time to critical events such as component degradation, and optimising maintenance strategies based on historical case analysis. Future research should focus on validating the model using real-world failure datasets with broader diversity, extending the approach to dynamic temporal models that reflect evolving technical conditions, and integrating the solution into IoT-enabled platforms for real-time equipment monitoring. Overall, the proposed model constituted a valuable contribution to the field of intelligent diagnostics, offering a balance between theoretical soundness and practical applicability, while its modular and adaptive structure allowed for extension to other classes of complex technical systems.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Amin, K., Kapetanakis, S., Polatidis, N., Althoff, K.-D., & Dengel, A. (2020). DeepKAF: A heterogeneous CBR & deep learning approach for NLP prototyping. In *2020 International conference on innovations in intelligent systems and applications (INISTA)* (pp. 1-7). Novi Sad. doi: 10.1109/INISTA49547.2020.9194679.

[2] Chen, M., Qu, R., & Fang, W. (2022). Case-based reasoning system for fault diagnosis of aero-engines. *Expert Systems with Applications*, 202, article number 117350. doi: 10.1016/j.eswa.2022.117350.

[3] Chen, Z., Xu, J., Alippi, C., Ding, S.X., Shardt, Y., Peng, T., & Yang, C. (2021). Graph neural network-based fault diagnosis: A review. *ArXiv*. doi: 10.48550/arXiv.2111.08185.

[4] Dubchak, L., Sachenko, A., Bodyanskiy, Y., Wolff, C., Vasylkiv, N., Brukhanskyi, R., & Kochan, V. (2024). Adaptive neuro-fuzzy system for detection of wind turbine blade defects. *Energies*, 17(24), article number 6456. doi: 10.3390/en17246456.

[5] Jimenez-Diaz, G., & Díaz-Agudo, B. (2024). Visualization of similarity models for CBR comprehension and maintenance. In *Case-based reasoning research and development (ICCBR 2024)* (pp. 67-80). Merida: Springer. doi: 10.1007/978-3-031-63646-2_5.
[6] Krüger, M. (2023). Comparative analysis of text-based CBR algorithms for cybercrime profiling investigations. In *Lernen, wissen, daten, analysen (LWDA) 2023* (pp. 359-371). Marburg: CEUR Workshop Proceedings.
[7] Leake, D., Ye, X., & Crandall, D. (2021). Supporting case-based reasoning with neural networks: An illustration for case adaptation. In *AAAI 2021 spring symposium on combining machine learning and knowledge engineering (AAAI-MAKE 2021)*. California: Palo Alto.
[8] Lin, N., Liu, H., Fang, J., Zhou, D., & Yang, A. (2023). An interpretability framework for similar case matching. *ArXiv*. doi: 10.48550/arXiv.2304.01622.
[9] Mathisen, B.M., Aamodt, A., Bach, K., & Langseth, H. (2020). Learning similarity measures from data. *ArXiv*. doi: 10.48550/arXiv.2001.05312.
[10] Mustyala, S., & Bisi, M. (2025). Ensembling harmony search algorithm with case-based reasoning for software development effort estimation. *Cluster Computing*, 28, article number 97. doi: 10.1007/s10586-024-04858-w.
[11] Neykov, N., & Stefanova, S. (2023). Using case-based reasoning in system diagnostics and maintenance. In *Proceedings of seventh international congress on information and communication technology* (pp. 345-359). Singapore: Springer. doi: 10.1007/978-981-19-2394-4_28.
[12] Ren, S., Gui, F., Zhao, Y., Zhan, M., & Wang, W. (2020). An effective similarity determination model for case-based reasoning in support of low-carbon product design. *Advances in Mechanical Engineering*, 12(10). doi: 10.1177/1687814020970313.
[13] Serrà, J., & Arcos, J.Ll. (2014). An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67, 305-314. doi: 10.1016/j.knosys.2014.04.035.
[14] Valdez-Ávila, M.F., Bermejo-Sabbagh, C., Diaz-Agudo, B., Orozco-del-Castillo, M.G., & Recio-Garcia, J.A. (2023). CBR-fox: A case-based explanation method for time series forecasting models. In *Case-based reasoning research and development. ICCBR 2023* (pp. 192-207). Cham: Springer. doi: 10.1007/978-3-031-40177-0_13.
[15] Virtanen, P., *et al.* (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272. doi: 10.1038/s41592-019-0686-2.
[16] Vychuzhanin, V., & Vychuzhanin, A. (2025). *Stochastic models and methods for diagnostics, assessment, and prediction of the technical condition of complex critical systems*. Lviv-Torun: Liha-Pres. doi: 10.36059/978-966-397-457-6.
[17] Xu, H., Wei, Y., Cai, Y., & Xing, B. (2023). Knowledge graph and CBR-based approach for automated analysis of bridge operational accidents: Case representation and retrieval. *PLoS ONE*, 18(11), article number e0294130. doi: 10.1371/journal.pone.0294130.
[18] Ye, J. (2017). Single-valued neutrosophic similarity measures based on cotangent function and their application in the fault diagnosis of steam turbine. *Soft Computing*, 21(3), 817-825. doi: 10.1007/s00500-015-1818-y.
[19] Ye, X., Leake, D., Wang, Y., Zhao, Z., & Crandall, D. (2024). Towards network implementation of CBR: Case study of a neural network K-NN algorithm. In *Case-based reasoning research and development. ICCBR 2024. Lecture notes in computer science* (pp. 354-370). Cham: Springer. doi: 10.1007/978-3-031-63646-2_23.
[20] Zeng, T., Bao, R., Qin, Y., Sun, X., Gao, Y., Cheng, L., Hou, P., Sang, H., Ma, L., & Zhou, X. (2025). MSFF-CBR: Case-based reasoning technology for adaptive multi-information fusion fault diagnosis. *Measurement Science and Technology*, 36(4), article number 045111. doi: 10.1088/1361-6501/adc474.
[21] Zuber, M., & Sirdey, R. (2021). Efficient homomorphic evaluation of k-NN classifiers. *Proceedings on Privacy Enhancing Technologies 2021*, 2, 111-129. doi: 10.2478/popets-2021-0020.

Адаптивна метрика оцінки схожості для інтелектуальної діагностики відмов у суднових енергетичних установках

Володимир Вичужанін

Доктор технічних наук, професор
Національний університет «Одеська політехніка»
65044, просп. Шевченка, 1, м. Одеса, Україна
https://orcid.org/0000-0002-6302-1832

Олексій Вичужанін

Доктор філософії, асистент
Національний університет «Одеська політехніка»
65044, просп. Шевченка, 1, м. Одеса, Україна
https://orcid.org/0000-0001-8779-2503

Анотація. Своєчасна та точна діагностика несправностей в суднових енергетичних установках (СЕУ) є надзвичайно важливою для забезпечення безпеки на морі, мінімізації експлуатаційних ризиків та оптимізації стратегій технічного обслуговування. Зі збільшенням складності систем, гетерогенності джерел даних та обмеженості історичних записів про несправності традиційні методи діагностики часто виявляються недостатніми, особливо в ситуаціях, пов'язаних з рідкісними або неоднозначними несправностями. Метою цього дослідження була розробка інтерпретованої, адаптивної та ймовірнісно обґрунтованої методології для оцінки схожості випадків несправностей у СЕУ для використання в інтелектуальних системах підтримки прийняття рішень. Запропонований метод інтегрує евклідову, джакардову та логістичну метрики схожості з байєсівським виведенням, моделюванням тимчасової деградації, корекцією ваги на основі частоти та контекстним згладжуванням уражених підсистем. Модель використовує оптимізацію L-BFGS-B для автоматичного коригування ваги метрик відповідно до діагностичної релевантності. Чисельні експерименти на основі синтетичних даних випадків показали високу точність класифікації: 96 % для відмов, пов'язаних з перегрівом системи охолодження, 84 % для випадків деградації підшипників і 92 % для порушень у подачі палива. Навіть при 40 % скороченні обсягу навчальних даних падіння продуктивності не перевищило 7 %, що свідчить про високу стійкість до розрідженості даних. Візуалізація меж прийняття рішень продемонструвала здатність моделі розрізняти перекривні класи відмов, зберігаючи семантичну інтерпретованість. Результати оптимізації ваги визначили «Тип відмови» як домінуючий фактор, тоді як «Категорія ризику» та «Уражені підсистеми» мали незначний вплив і були виключені. Байєсівське агрегування ще більше підвищило надійність діагностичних висновків, поєднавши локальну схожість із глобальними статистичними апріорними даними. Розроблена методологія може бути ефективно застосована морськими інженерами, операторами суден та розробниками інтелектуальних діагностичних платформ для виявлення несправностей, аналізу першопричин та прогнозного технічного обслуговування в умовах невизначеності та неповної інформації. Її модульна структура також дозволяє розширити її застосування на інші складні технічні галузі, крім СЕУ

Ключові слова: система підтримки прийняття рішень; байєсівська агрегація; адаптивні вагові коефіцієнти; прецедентний аналіз відмов; імовірнісна метрика; схожість технічних станів

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.852:519.23
DOI: 10.62660/bcstu/3.2025.93

Study of the impact of different categorical feature encoding techniques on cluster structures

Natalia Kondruk*
PhD in Technical Sciences, Associate Professor
Uzhhorod National University
88000, 14 Universytetska Str., Uzhhorod, Ukraine
https://orcid.org/0000-0002-9277-5131

Inna Neroda
Postgraduate Student
Uzhhorod National University
88000, 14 Universytetska Str., Uzhhorod, Ukraine
https://orcid.org/0009-0009-3686-3565

Abstract. Categorical features are a common type of data used in data analysis, but their non-metric nature makes it difficult to apply standard clustering algorithms. The relevance of the study is conditioned by the need to assess the impact of different methods of recoding (digitisation) of such features on the effectiveness of cluster analysis. The purpose of the study was to investigate how different techniques of categorical data processing affect the quality and structure of clusters. The methodology included the implementation of three models with different approaches to variable coding: without taking into account domain specifics, considering the content of the features, and with alternating the order of application of clustering and dimensionality reduction approaches. LabelEncoder, OrdinalEncoder, One-Hot Encoding, Mapping, and MultiLabelBinarizer were used for coding. In each of the models, clustering was performed using two algorithms – K-Means and agglomerative clustering, which allowed comparison of their sensitivity to changes in data representation. The t-SNE dimensionality reduction method was used to visualise the cluster structure in two-dimensional space. The quality of clustering was evaluated using the Silhouette Score, Dunn Index, Davies-Bouldin Index, and Calinski-Harabasz Index metrics. The data for the analysis were obtained from an open source and contained information about the psycho-emotional state of students. The study found that the basic recoding of categorical features without considering their semantics and context negatively affected the quality of clustering, reducing the accuracy of the division and complicating the interpretation of the results. Instead, the use of domain-oriented coding approaches ensured the development of clusters with clearer boundaries and a more logical internal structure. In addition, it was found that changing the sequence of clustering and dimensionality reduction affects the preservation of local relationships in the data. It was analysed that different approaches change both the number and quality of clusters, which was reflected in the values of the evaluation metrics. The practical significance of the results lies in the possibility of their application by data analysts and machine learning specialists to improve the accuracy of segmentation of complex categorical data

Keywords: data analysis; machine learning; unsupervised learning; automatic object grouping; segmentation

Article's History: Received: 06.05.2025; Revised: 11.08.2025; Accepted: 15.09.2025.

Suggested Citation:
Kondruk, N., & Neroda, I. (2025). Study of the impact of different categorical feature encoding techniques on cluster structures. *Bulletin of Cherkasy State Technological University*, 30(3), 93-105. doi: 10.62660/bcstu/3.2025.93.

*Corresponding author

INTRODUCTION

Most of the real data contains categorical features that characterise the qualitative properties of objects, such as colour, product type, geographical location, level of education, or disease category. Unlike numerical variables, which have a natural order and clear metric interpretation, categorical data does not have a numerical equivalent or internal metric structure, which makes it difficult to use them directly in standard clustering algorithms. Most clustering algorithms operate in numerical spaces by applying distance metrics or similarity measures that cannot be directly applied to categorical features. To overcome this problem, a number of coding techniques have been developed that allow transforming categorical variables into numerical representations. Among the most common methods are one-hot encoding, which converts categories to binary vectors, and label encoding, which assigns each category a unique numeric code.

Choosing a coding technique is not a trivial task, as it can significantly affect the results of clustering. It is the way categorical variables are represented that determines how the algorithm interprets distances or similarities between objects, and therefore, how clusters will form. Accordingly, there is a need for a systematic study of various techniques for encoding categorical variables and their impact on cluster development. This includes analysing how different coding approaches change the distribution of objects in clusters, affect their compactness and separability.

Studies confirm that the choice of categorical feature coding technique significantly affects the quality of clustering and the interpretability of the resulting groups. Thus, M. Anitha *et al.* (2025) proposed a new Chi-Square target Encoding (CSTE) approach that uses chi-square statistics to estimate the association between categories and the target variable. This method allows efficient storage of information during the transformation of categorical features into numerical representations, which positively affects the accuracy of clustering. CSTE demonstrated an advantage over conventional methods such as one-hot encoding and fuzzification, achieving an average accuracy gain of 3.94%, and high F1 and AUC values for various data sets.

Similarly, Z. Liang (2025) proposed new encoding methods for highly cardinal categorical variables, in particular, low-level encoding and logistic regression, which allows reducing the dimension of data without losing important characteristics. These methods provide compact and informative feature vectors that help to improve clustering efficiency and reduce the risk of overtraining. The study showed significant improvements in model performance and computational efficiency compared to conventional methods such as one-hot encoding, especially when working with large data sets.

In the field of medical research, H. Hafid & S. Annisa (2025) applied K-Medoids and K-Prototypes techniques to cluster hypertensive patients, demonstrating the importance of successfully selecting categorical variable encoding for accurate grouping of medical data. The researchers noted that the use of the K-Prototypes algorithm, which combines clustering of numerical and categorical features, provides a more adequate development of patient groups compared to conventional methods based only on numerical data. The researchers emphasised that the correct choice of categorical variable coding techniques significantly affects the accuracy and stability of clustering, which is critical for medical applications where segmentation accuracy directly affects the quality of diagnosis and treatment.

Clustering as a method for identifying groups in data largely depends on how categorical features are transformed into numeric spaces. According to A.M. Ikotun *et al.* (2023), the K-Means algorithm remains one of the most common clustering methods, especially in big data environments. However, its efficiency is significantly reduced when working with categorical variables without proper transcoding, which can lead to distortion of the cluster structure and reduce the accuracy of the results. E.K. Tokuda *et al.* (2022) concluded in their paper that agglomerative clustering methods, in particular, those based on different merge criteria (single, average, median, complete, centroid, and Ward), demonstrate flexibility in working with different data types. However, these methods are also prone to detecting false-positive clusters, especially in conditions of homogeneous data distributions. This highlights the importance of carefully choosing categorical feature encoding methods, since incorrect transformation can lead to the development of clusters that do not reflect the actual data structure. However, the literature does not pay enough attention to the direct impact of coding techniques on the cluster structure, which leaves room for further research in this area.

An additional aspect related to preparing data for clustering concerns dimensionality reduction. According to N.E. Kondruk (2023), the use of dimensionality reduction techniques is an important prerequisite for improving the efficiency of machine learning models. The researcher considered several techniques, in particular, Principal Component Analysis (PCA), Isometric Mapping (Isomap), Uniform Manifold Approximation and Projection (UMAP), and t-SNE, and concluded that t-SNE provided the best predictive results for the problem of classifying high-dimensional data and that the use of advanced methods of reducing the dimension of data can contribute to the construction of more efficient models.

Overall, research confirms that conventional methods such as one-hot encoding and label encoding have limitations, especially when working with large data sets. New approaches that consider statistical relationships between categories can improve the quality of clustering and reduce overlap between groups. The purpose of this study was to conduct a comparative analysis of the influence of various techniques for encoding categorical features on the quality and structure of clusters in machine learning problems.

This study examined how different approaches to encoding categorical data can change the distribution of objects in clusters and which methods provide the best performance in different data contexts. In addition, the application of various clustering methods and clustering quality indices was considered to evaluate the effectiveness of different techniques for encoding categorical features using the example of segmenting students according to their psycho-emotional health.

MATERIALS AND METHODS

Clustering was evaluated using internal, external, and graphical methods. Internal methods were based on the characteristics of the data itself and did not require external information. One of the main indicators was the silhouette coefficient, which evaluated the quality of clustering by analysing the grouping of objects within clusters and their isolation from other clusters. The value varied from -1 to 1, where higher values indicated better cluster separation (Kondruk, 2019). In addition, the Davies-Bouldin index was used, which measured the average similarity between clusters and their resolution (lower values indicate better clustering) (Wegmann *et al.*, 2021). The Akaike Information Criterion (AIC) helped to assess the complexity of the clustering model: lower values indicate a better balance between the quality of clustering and the number of model parameters (Sánchez Vinces *et al.*, 2025).

External methods considered comparisons with reference distributions or correct cluster labels. Graphical methods included a dimensionality reduction procedure for visual evaluation of clusters. The PCA method was previously tested, but due to the nonlinear nature of the data and the lack of linear relationships in the dataset under consideration, it was not applicable. Instead, the nonlinear t-SNE method was used, which better preserves local distances between points and allowed estimating the density and overlap of clusters in two-dimensional space.

The K-Means method is a clustering method designed to split data into k clusters. Its purpose was to split data into a predefined number of clusters (K) so that objects within one cluster have a greater similarity to each other than objects in other clusters (Lloyd, 1982). Agglomerative clustering is a hierarchical analysis method that begins with each object being treated as a separate cluster. The algorithm gradually combines the two nearest clusters at each stage, until all objects form one large cluster. This approach allowed investigating the data structure and visualising relationships between objects (Miyamoto, 2022; Sieranoja & Fränti, 2025; Behzadidoost & Izadkhah, 2025).

For experiments, data on the psychoemotional health of students were taken. Data set based on the survey by V.A. Ashfaq (n.d.), consisted of 87 entries and 21 attributes and contained information about students, their students, and their educational experience, psychological state, and social aspects of life. The study was conducted in accordance with the ethical principles set out in the Declaration of Helsinki (World Medical Association's Declaration of Helsinki, 2013).

In this regard, the study focused on three different models for transcoding categorical variables, each of which offered a unique approach to transforming the original data. Model 1: data was transcoded without considering their nature, which allowed creating a neutral representation of variables for further analysis, followed by data compression and clustering. Model 2: data transcoding was performed considering feature domains, which allowed more accurately reflecting their essence and ensuring correct representation, then data compression and clustering occurs. Model 3: data was transcoded based on its nature, then the data was clustered, and only then the dimensionality reduction method was applied for further visualisation.

In Model 1, label encoder, ordinary encoder, and mapping methods were used to process categorical features, depending on their nature. This approach allowed getting a neutral representation of the source data and minimising the impact of subjective assumptions about their semantics. The label encoder method was used to transcode columns such as "gender", "degree_level", "residential_status", and "campus_discrimination". This method converted attributes to numeric values, while maintaining easy transcoding. It was ideal for signs where there is no natural order, such as gender, educational level, and others. Based on this, the method minimised the complexity of analysis, ensuring ease of working with data. The ordinary encoder method was applied to the "university" and "degree_major" columns. This approach allowed organising categories in a numeric format, which can be important for maintaining the order of values. For example, in the case of universities and educational specialities, the method allowed comparing their unique values with numbers for more convenient further processing.

Category mapping using the dictionary was used for the "academic_year", "cgpa", "sports_engagement", "average_sleep", and "stress_relief_activities" columns. This approach allowed manually mapping text values to numeric values, which was especially convenient for

displaying categories with a natural sequence or simplifying multidimensional data into a single numeric form. For example, in the "academic_year" column, the sequence of academic years is represented in numerical form to maintain a logical order, and the ranges of GPA scores in the "cgpa" column were displayed in numerical averages. Similarly, the degrees of activity in sports, sleep hours, and relaxation activities were transcoded in such a way as to preserve the content of information and increase the convenience for further analysis. Categorical variables were transcoded to numeric format, and the t-SNE method was used to reduce the dimension of the feature space and visualise the data structure.

To analyse clustering using the K-Means method, key metrics were used to assess its quality and determine the optimal number of clusters. Each metric had its own meaning in the decision-making process. Silhouette Score was used to determine the degree of isolation and compactness of clusters. Davies-Bouldin Score allowed estimating the degree of overlap between clusters, where smaller values indicated a clearer separation. The AIC criterion was used to select the model that best balances complexity and clustering quality.

Agglomerative clustering considered the hierarchical structure of data and helped to better identify complex relationships, which was especially noticeable in cluster visualisation. Clustering quality assessment was performed using metrics such as Silhouette Score to assess cluster compactness, and AIC to select the optimal balance between model accuracy and complexity. In addition, a dendrogram was used for agglomerative clustering, a graphical tool that displays the hierarchical process of combining points. The number of clusters that best reflected the data structure can be determined by drawing a horizontal line at a certain distance level.

In Model 2, to ensure the correct representation of data, various transcoding methods were applied for each variable, considering their nature and characteristics. For variables that had a limited set of categories, such as "gender" and "residential_status", the method of simple transcoding using label encoder was used. This approach allowed efficiently representing nominal categories in numeric format, while maintaining their nominal nature without creating additional columns. For variables with an extended set of unique values, such as "university" and "degree_major", the binary encoding method was used – one-hot encoder. This method allowed creating separate binary columns for each category, which avoided logical priority between them and preserved their uniqueness. Additional created columns extended the data set while maintaining its full interpretation.

Variables with ranges of values or ordered categories (academic_year, cgpa, sports_engagement, average_sleep) were transformed using mapping. This allowed converting text ranges to numeric levels or the average value of the range. For example, a scale was created for the CGPA variable that displays the average value of each range, and for "sports_engagement" – numerical levels according to the frequency of sports activities. The "stress_relief_activities" variable contains multiple categories for each respondent, so the "multilabelbinariser" technique was used to process it. This allowed presenting respondents' participation in several activities simultaneously in a binary matrix format. This encoding format allowed considering potential relationships between activities in further analysis. To assess the quality of clustering using the K-Means method, the Silhouette Score and Kalinski-Harabasz Score indices were used, which allowed determining the degree of compactness and resolution of clusters. Similarly, the Calinski-Harabasz index was used for agglomerative clustering, and a visual study of the dendrogram was performed to determine the optimal separation threshold and the number of clusters.

In Model 3, data was transcoded based on the nature of each variable, just as in Model 2. The main difference between these models was the sequence of stages. In Model 2, data was first transcoded, then its dimension was compressed, and only then clustering was performed. This reduced the dimension of data even before clusters were formed, which can simplify the clustering process, but may affect the preservation of the original data structure. In Model 3, by contrast, data transcoding was accompanied by clustering before dimension compression. This change in the sequence allowed focusing on the development of clusters in their original form, and only then applying dimensionality reduction methods to visualise the resulting structures.

To determine the optimal number of clusters in the K-Means method, a sequential analysis of the values of the Silhouette Score, Dunn Index, and Davies-Bouldin index metrics was performed on different cluster numbers. Evaluation of these metrics allowed selecting the number of groups that provided a balance between compactness and cluster resolution in the data under study. In the case of agglomerative clustering, in addition to calculating the Silhouette Score, a visual study of the dendrogram was performed. Changes in cluster structure at different cut-off thresholds were analysed to find the number of groups that best reflects the natural divisions in the data hierarchy. The selected separation level was supposed to provide maximum clarity for clusters without excessive merging or splitting.

RESULTS AND DISCUSSION

After digitising the categorical data by transcoding them, a correlation matrix was constructed for preliminary analysis of the relationships between features, which is shown in Figure 1.

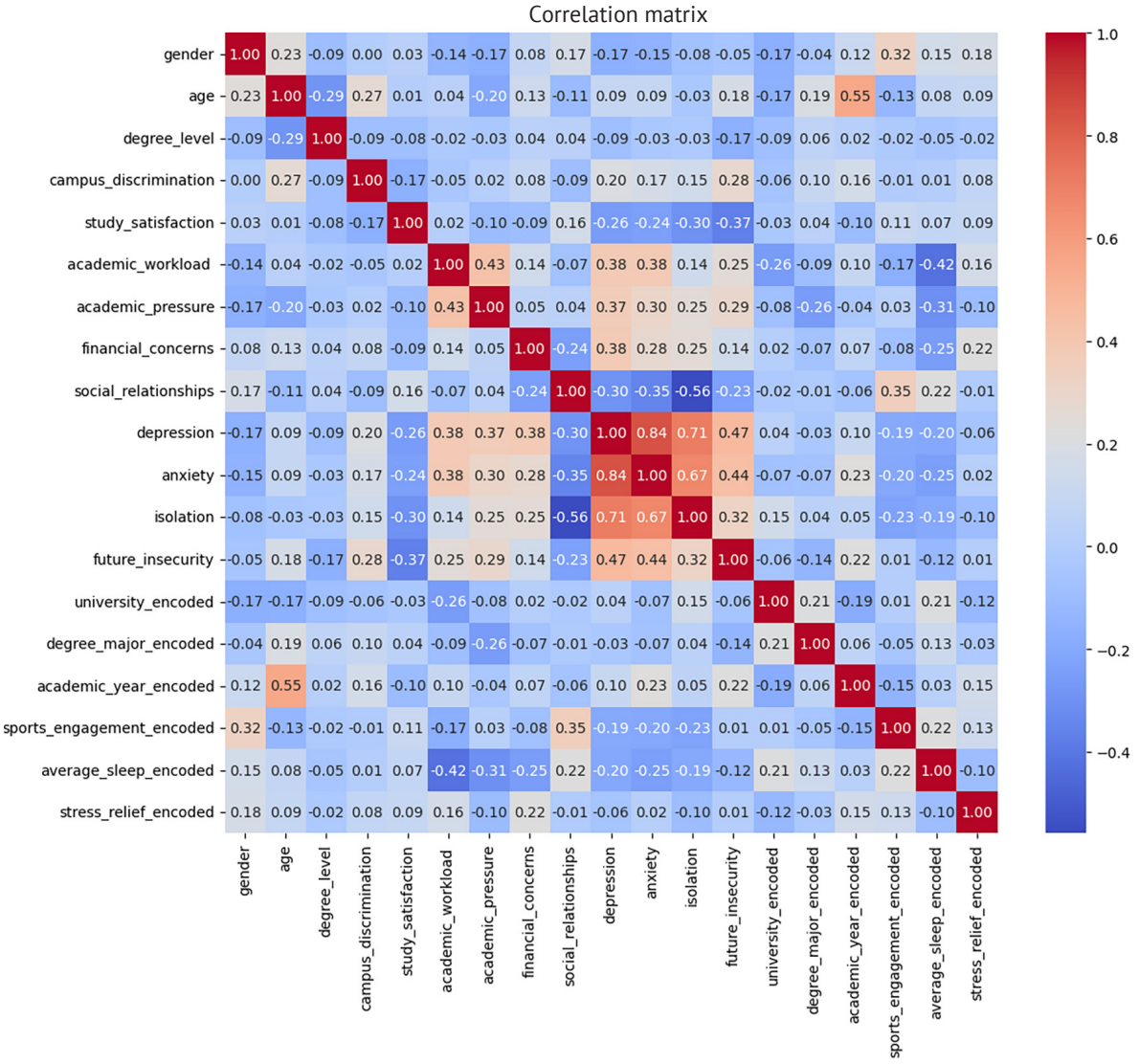


Figure 1. Correlation matrix

Source: compiled by the authors

The analysis showed that the data has a predominantly nonlinear structure, which influenced the choice of further processing methods. To reduce the data dimension, the PCA method was initially considered, but due to inconsistencies with the assumptions of this method (in particular, the lack of multicollinearity), the t-SNE algorithm was chosen, which better reflects local structures and is able to detect nonlinear dependencies in the data.

Clustering results based on Model 1. The K-Means method with 6 clusters demonstrates the division of data into groups characterised by moderate compactness and separation. The Silhouette Score (0.4) reaches a local maximum for six clusters, which indicates the best combination of cluster compactness and distance between them. A low Davies-Bouldin Score (0.8) confirmed that the clusters have a clear structure with minimal overlap. The AIC criterion, which is 804, states that the choice of six clusters provides an optimal balance between the accuracy of the model and its complexity. Despite a certain degree of data noise and overlap, the K-Means method effectively revealed basic patterns and structures that allow data to be divided into groups with relatively clear boundaries.

Analysis of the results of agglomerative clustering for six clusters shows that the algorithm successfully grouped data in a multidimensional space, which is confirmed by the distribution of points in the visualisation. Silhouette Score has a local maximum of six clusters, which indicates the best compactness and clarity of separation. The AIC graph shows the minimum value for six clusters, which confirms the optimal balance between the accuracy of the model and its complexity. The clusters are clearly separated and show moderate compactness. For example, well-defined groups (cluster

0, 1) indicate a clear structure, but some clusters (such as 2 and 5) have more overlap, reflecting more complex relationships in the data. The dendrogram shown in Figure 2, used as an additional tool, clearly reflects the hierarchical process of cluster development and allows confirming the choice of six clusters as the optimal division that preserves the natural structure and relationships between objects. To visualise the spatial distribution of clusters, the t-SNE dimensionality reduction method was used (Fig. 3). This approach allowed for visual assessment of the structure and isolation of clusters in two-dimensional space, which confirms the results of clustering and helps to better understand the internal relationships between groups.

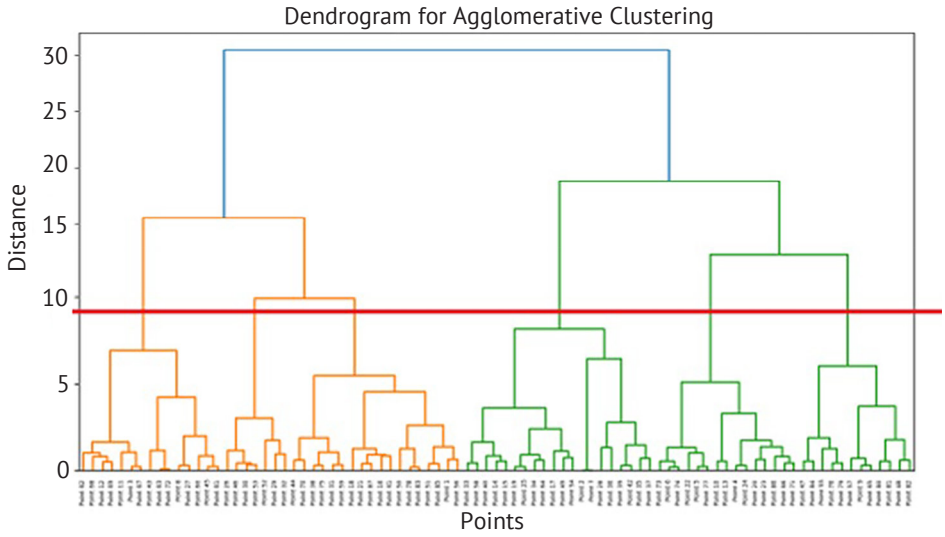


Figure 2. Dendrogram for Model 1

Source: compiled by the authors

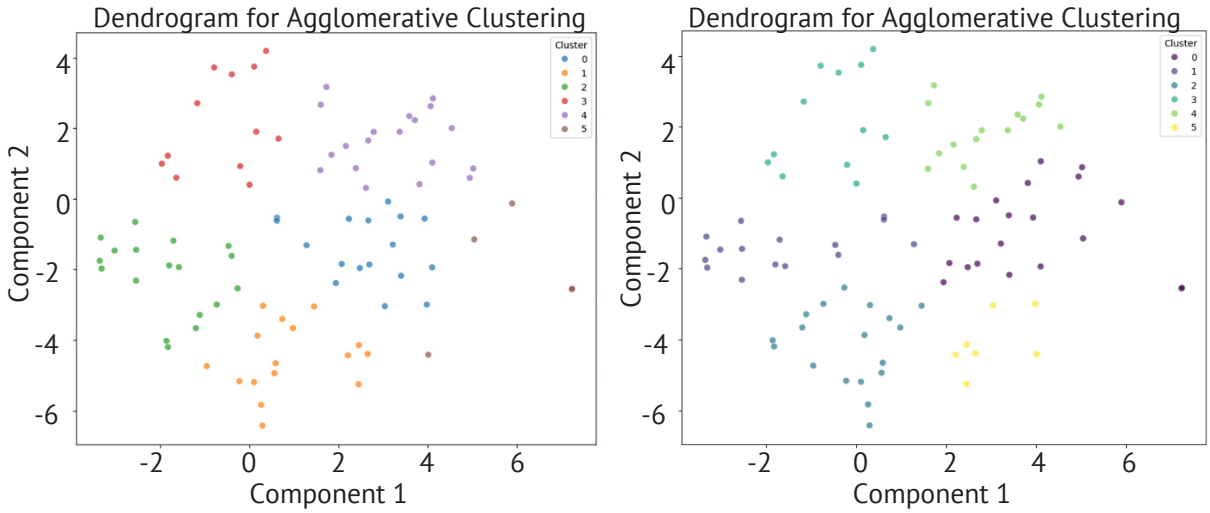


Figure 3. Clustering results for Model 1

Source: compiled by the authors

Clustering results based on Model 2. The results of clustering using the K-Means method with six clusters demonstrated a clear grouping of most data with high compactness and minimal mixing between clusters. Analysis of the Silhouette Score and Kalinski-Harabasz Score shows that the local maximum of the former and the high value of the latter are achieved in six clusters. This indicates optimal compactness and separation of clusters by this number. Thus, six clusters are the optimal choice, since they provide maximum clustering quality.

For agglomerative clustering, analysis of the Calinski-Harabasz Index demonstrated that the maximum ratio between inter- and intra-cluster variance is achieved in five clusters, which indicates an optimal separation of groups with high internal compactness and maximum distance between them. In addition, the dendrogram shown in Figure 4 was used as an additional visual tool that confirmed the selection of five clusters, demonstrating a clear separation of independent groups at the established cut-off threshold.

Thus, the choice of five clusters was consistent with both numerical metrics and visual analysis of the hierarchical data structure. This distribution indicates the presence of homogeneous subgroups with characteristic features, which is important for further analysis. Clear boundaries between clusters ensure the stability of groups and provide grounds for practical interpretation of their characteristics. Thus, the results of agglomerative clustering confirm the mathematical validity of the choice of five clusters and the correspondence of the breakdown to the structure of the original data (Fig. 4).

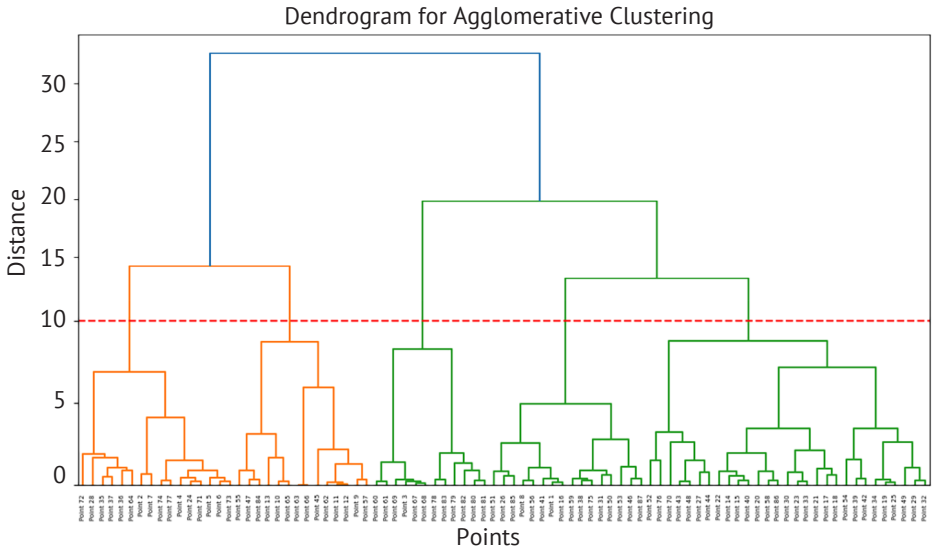


Figure 4. Dendrogram for Model 2

Source: compiled by the authors

The results of numerical and visual analysis confirm the feasibility of the selected clustering parameters for Model 2. Figure 5 shows visualisations of cluster distributions that illustrate the data structure and segmentation quality obtained by K-Means and agglomerative clustering methods.

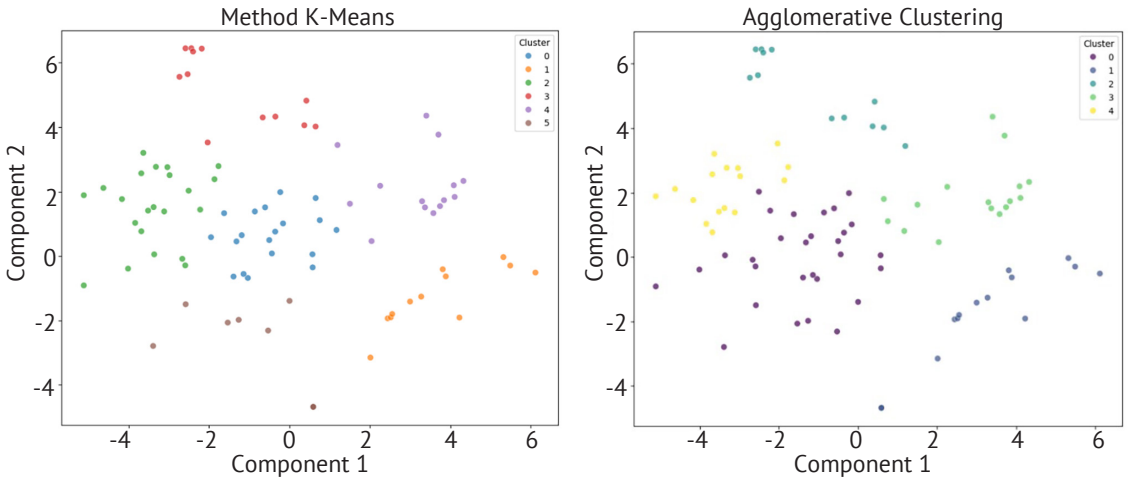


Figure 5. Clustering results for Model 2

Source: compiled by the authors

For the K-Means method, there is a clear development of six clusters, each of which has a high compactness and is well separated from the others. Minimal overlap between some groups (in particular, clusters 2 and 4) indicates the complexity of the source data structure, but the overall segmentation looks logical and stable. In the case of agglomerative clustering, segmentation into five clusters was optimal, which show better separation between groups compared to Model 1. This suggests that consideration of the content of features in encoding allows more accurately preserving the natural data structure even in a complex multidimensional space.

Clustering results based on Model 3. For the K-Means clustering method, it is important to determine the optimal number of clusters that will ensure high-quality data segmentation. The Silhouette Score Index reached a local maximum in six clusters, which indicates high internal cohesion and clear data structure in each cluster. The Dunn Index also showed high values for six clusters, indicating the compactness of the clusters and their good separation. The Davies-Bouldin Index reached a minimum value with the same number of clusters, which indicates low inter-cluster similarity and clustering quality in general. Therefore, the choice of six clusters for K-Means is optimal, since it provides a balance between internal cohesion and group separation. Cluster overlap indicates that some points lie at the boundary between groups, which is natural for complex data with a continuous structure. In general, clustering still allows isolating the main data structure and ensuring their overall distribution into groups.

For agglomerative clustering, the Silhouette Score Index reaches a local maximum for four clusters, which indicates an optimal balance between cohesion within clusters and their separation. The dendrogram analysis shown in Figure 6 confirmed this choice – when divided into four groups, the data form well-defined clusters with a distance between them of about 15 units, which minimises overlap. Fewer clusters lead to less detailed segmentation and loss of local patterns.

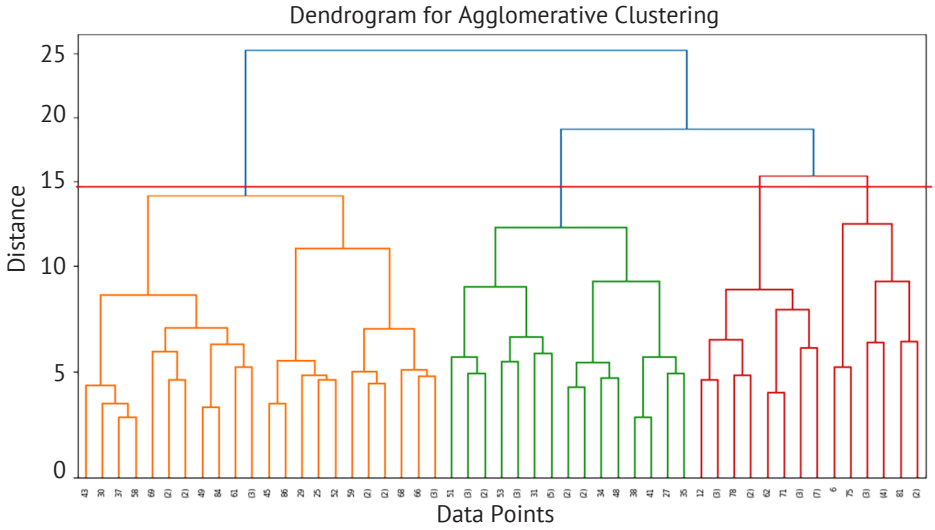


Figure 6. Dendrogram for Model 3

Source: compiled by the authors

To further confirm the qualitative characteristics of clustering using Model 3, the resulting clusters were visualised. The data distributions shown in Figure 7 using the K-Means method (6 clusters) and agglomerative clustering (4 clusters) allow estimating the spatial structure of groups, their compactness, the degree of separation, and the presence of overlapping zones.

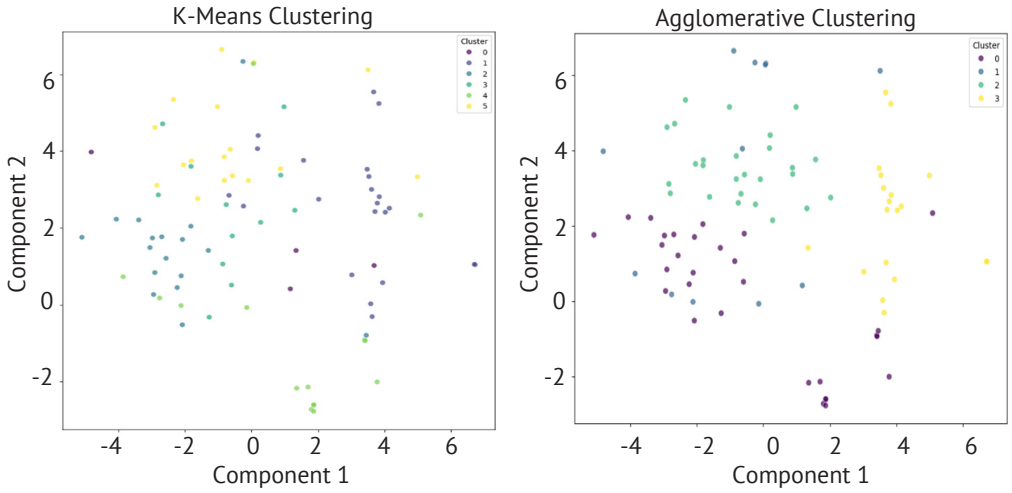


Figure 7. Clustering results for Model 3

Source: compiled by the authors

Analysis of Model 1 showed that the optimal number of clusters for both clustering methods is six, which was confirmed by quality assessment metrics. For K-Means, metrics indicate acceptable clustering quality, although data noise affected the clarity of boundaries between individual groups. Agglomerative clustering allowed isolating six clusters using a dendrogram. The clarity of the distribution largely depended on the cut-off level, which confirms the influence of the hierarchical structure on clustering results. Within this model, the method provided adequate data segmentation, although partial overlap between certain clusters was observed. Overall, the results of Model 1 showed that clustering methods provided an adequate data structure, but the choice of encoding method affects the accuracy of segmentation and the clarity of cluster boundaries. This highlights the importance of further research on Models 2 and 3, which can eliminate the limitations of Model 1 by more accurately considering the nature of variables or changing the sequence of analysis steps.

Model 2 showed improved clustering accuracy by considering feature domains when transcoding categorical features. This approach ensures correct representation of features, which significantly affects the quality of subsequent stages of analysis. As a result, clustering becomes more structured, with clear boundaries between groups, which confirms the effectiveness of data preparation in this model. The K-Means clustering method allowed creating six compact clusters that show a clear grouping of most data. A slight overlap between some clusters (for example, between 2 and 4) indicates data complexity, but the overall result of clustering is balanced and logical. The model provided efficient segmentation by identifying natural patterns in the data. Agglomerative clustering also showed a noticeable difference: in Model 2, five distinct groups were isolated with less overlap due to the domain nature of the features. This allowed better isolating the data while maintaining its uniformity. In Model 1, partial mixing between groups was more pronounced, which reduced the accuracy of the results. Thus, consideration of feature domains in Model 2 not only improved the quality of clustering, but also made data segmentation more logical and consistent with their natural structure. This highlights the importance of correctly encoding variables to achieve high-quality analysis.

Analysis of Model 3 showed that the sequence of stages – allowed focusing on clusters that best correspond to the original data structure. This approach provides clear grouping while preserving important information about variables. The results of clustering using the K-Means method show the development of six clusters. Although cluster overlap is observed in multiple zones, this effect is due to the natural complexity of the data. Despite this, the method provided high-quality grouping of points into compact groups, which indicates the effectiveness of segmentation and identification of patterns. Agglomerative clustering distinguishes four clusters with a well-structured data distribution. Most clusters have clear boundaries and show a high level of internal cohesion. Small areas of overlap indicate similarity of characteristics between neighbouring groups, but the overall segmentation is clear and logical, which confirms the strength of the method in analysing complex data. Thus, Model 3 provides reliable segmentation of data, helping to preserve its natural structure. This highlights the effectiveness of selecting a sequence of analysis steps and the importance of considering the nature of variables when building a clustering model.

This study demonstrated that a flexible and semantically sound approach to coding – considering the nature of features, their hierarchy, scale, and domain value – can improve data segmentation in clustering tasks. Processing categorical variables is a critical step for classification, clustering, and data analysis tasks. As shown in a large-scale study by F. Matteucci *et al.* (2023), which covered 32 coding methods (including One-Hot, ordinary, Mean-Target, GLMM, Min-Hash, CatBoost, and Sum), the choice of encoder has a significant impact on the quality of results, and its effectiveness depends on a combination of experimental factors: model type, quality metrics, hyperparameter tuning strategy, and how results are aggregated. This highlights the need for a flexible and meaningful approach to coding that considers the nature of features, their hierarchy, and task specifics. However, instead of fully automated solutions, such as the CESAMO method (Valdez-Valenzuela *et al.*, 2024), which automates statistically optimised category coding, retained control over correspondence between categories and their numerical representations. This provided greater transparency and control over the interpretation of results, which is especially important in a limited data set. The importance of adapting coding methods to the characteristics of data was confirmed in research conducted by D. Breskuvienė & G. Dzemyda (2023), who analysed the impact of various coding techniques on classification quality in problems of detecting fraudulent transactions with highly unbalanced categorical data. The researchers pointed out that targeted encoders, such as Weight of Evidence and James-Stein, can significantly improve the performance of models, but the effectiveness of individual methods, such as CatBoost, decreases with a strong imbalance, which emphasises the need to choose an encoding based on the specifics of the task and data structure.

Confirmation of the importance of an adaptive approach to working with categorical variables can be found in the review paper by T. Dinh *et al.* (2024), devoted to the evolution of categorical data clustering methods over the past 25 years. The researchers noted that successful clustering of categorical variables requires not only the correct choice of algorithm, but also careful pre-coding, considering the nature of the variables, otherwise there is a risk of losing important semantic

connections. This approach is fully consistent with the approach in this paper, which combined different transcoding methods depending on the feature content and data logic. Preservation of category semantics allowed for more structured and interpreted clustering.

The problem of clustering categorical and mixed data types was also actively considered in many studies of the last 5 years. In particular, C. Di Nuzzo (2024) focused on improving spectral clustering for such data types by developing new nuclear functions that better consider the specifics of categorical relationships. The paper showed that standard Euclidean approaches may not be sufficient to adequately map internal structures in mixed sets, which is consistent with the observations in this study. For example, in Model 3 in this study, the use of clustering before the dimension reduction step contributed to the preservation of the natural feature structure. Although the researcher focused on spectral clustering, the results highlighted the importance of adapting clustering methods to the nature of variables – an approach that was also followed in model development.

In this context, it is also worth mentioning the study by D. Soemitro & J.F.S.R. Neto (2024), where an alternative approach to spectral clustering of categorical and mixed data was proposed. The researchers avoided conventional preprocessing, particularly one-hot or multi-binary, and instead suggested adding additional nodes to the graph corresponding to feature categories. This approach allowed considering both numerical and categorical information without losing the semantics of features, and also provided interpretability of clustering. This is especially true when using the MultiLabelBinarizer technique, which, although convenient, can lead to excessive dimension and loss of links between categories. The approach proposed by the researchers demonstrated the competitive quality of clustering and, importantly, allowed achieving linear complexity for the case of purely categorical data. Their study reinforced the argument for adaptive method selection based on data type and feature structure.

The choice of encoding method for categorical variables is crucial for the effectiveness of machine learning models, especially in the case of variables with a large number of levels. In a large benchmark study by F. Pargent *et al*. (2022) showed that target encoding, in particular, its regularised versions, provides better predictive results compared to conventional approaches such as ordinal or one-hot encoding. This confirms the importance of choosing an informed coding method based on the structure of variables, which is consistent with the approach to combining multiple coding strategies according to the nature of features.

H.L. Smith *et al*. (2024) drew attention to the potential distortions that can occur when using target encoding in combination with tree models. The researchers have shown that when using target-based coding to construct an ensemble of out-of-bag evaluation (OOB), they can significantly underestimate the classification error and overestimate the importance of variables. These results highlighted the limitations of target coding and the importance of step-by-step control of the feature processing process, an aspect that was considered when designing models and determining the sequence of their processing.

Another important aspect was the consistency between the choice of coding method and the type of machine learning model. A recent study by W. Zhu *et al*. (2024) contains both theoretical and empirical analysis of 14 methods for encoding categorical variables in combination with various types of models. The researchers proved that one-hot encoding is optimal for models that perform affine transformations (for example, multilayer perceptrons), while target encoding and its variants are best suited for models based on decision trees (for example, Random Forest). These results are fully consistent with the observations obtained from this study, that the correspondence between the coding method and the model type significantly affects the final result.

Thus, the approach of this study was based on proven methods of processing categorical variables and at the same time adapts the latest concepts, such as contextual coding, combined coding by variable type and adaptive order of processing stages. It was confirmed that there is no universal method: the effectiveness depends on both the type of problem and the nature of the features. The results obtained coincide with trends in data analysis.

CONCLUSIONS

In the course of the study, a detailed analysis of three different clustering models was carried out, which differ in approaches to processing and transcoding input categorical data. The results showed that the quality, structure, and clarity of the developed clusters largely depend on the chosen data preparation methods, and on the sequence of their application. This highlights the importance of carefully selecting and combining methods to achieve optimal results in segmentation problems.

Model 1, in which categorical features were transcoded without considering their domain nature, provided basic data separation into six clusters for both K-Means and agglomerative clustering. However, the fuzzy boundaries between clusters and partial overlap indicate that this approach is limited in displaying a complex data structure. This indicates that the method is not sufficiently adapted to the features of categorical variables, which negatively affects the accuracy of segmentation and reduces the quality of clustering in general.

Model 2, in which the transcoding of categorical variables was carried out considering their domain nature, significantly improved clustering results. In this model, six distinct clusters were obtained using the

K-Means method and five clusters using agglomerative clustering. The improvement in clustering quality was confirmed by higher metric values and reduced overlap between groups. This result demonstrated the effectiveness of correct transcoding, which better reflects the internal structure of multidimensional data and contributes to more accurate and logical segmentation.

Model 3, which assumed a change in the sequence of steps – performing clustering before reducing the dimension – showed the best ability to preserve the natural data structure. The K-Means method identified six clusters, while agglomerative clustering identified four clearly structured clusters. A small overlap in some areas does not prevent the preservation of important relationships between features, which is necessary for analysing complex multidimensional sets. This approach highlighted the importance of the correct sequence of analytical steps to preserve key data characteristics.

The results of the study confirmed that the choice of approach to transcoding categorical variables, and the number of clusters, directly affect the cluster structure of data. Different coding strategies can form a different number of clusters, and therefore, create fundamentally different segmentation schemes. Thus, the correct choice of encoding method, considering the domain specifics of features, and the sequence of processing are critical factors for achieving high clustering quality. The results obtained open up prospects for further improvement of algorithms for segmenting complex data and their application in various applied fields – from marketing and healthcare to social research and bioinformatics.

ACKNOWLEDGEMENTS
None.

FINANCING
None.

CONFLICT OF INTEREST
None.

REFERENCES
[1] Anitha, M., Savarimuthu, N., & Bhanu, S. (2025). Chi-square target encoding for categorical data representation: A real-world sensor data case study. *SN Computer Science*, 6, article number 228. doi: 10.1007/s42979-025-03766-z.

[2] Ashfaq, V.A. (n.d.). *Student mental health survey*. Retrieved from https://www.kaggle.com/datasets/abdullahashfaqvirk/student-mental-health-survey/data.

[3] Behzadidoost, R., & Izadkhah, H. (2025). Identifying effective algorithms and measures for enhanced clustering quality: A comprehensive examination of arbitrary decisions in hierarchical clustering algorithms. *Journal of Classification*, 42, 457-489. doi: 10.1007/s00357-025-09506-5.

[4] Breskuvienė, D., & Dzemyda, G. (2023). Categorical feature encoding techniques for improved classifier performance when dealing with imbalanced data of fraudulent transactions. *International Journal of Computers Communications & Control*, 18(3). doi: 10.15837/ijccc.2023.3.5433.

[5] Di Nuzzo, C. (2024). Advancing spectral clustering for categorical and mixed-type data: Insights and applications. *Mathematics*, 12(4), article number 508. doi: 10.3390/math12040508.

[6] Dinh, T., Hauchi, W., Fournier-Viger, P., Lisik, D., Ha, M.-Q., Dam, H.-C., & Huynh, V.-N. (2024). Categorical data clustering: 25 years beyond K-modes. *ArXiv*. doi: 10.48550/arXiv.2408.17244.

[7] Hafid, H., & Annisa, S. (2025). Implementation of K-medoids and K-prototypes clustering for early detection of hypertension disease. *Barekeng: Journal of Mathematics and Its Application*, 19(1), 465-476. doi: 10.30598/barekengvol19iss1pp465-476.

[8] Ikotun, A.M., Ezugwu, A.E., Abualigah, L., Abuhaija, B., & Heming, J. (2023). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622, 178-210. doi: 10.1016/j.ins.2022.11.139.

[9] Kondruk, N.E. (2019). A comparative study of cluster validity indices. *Radio Electronics, Computer Science, Control*, 4, 59-67. doi: 10.15588/1607-3274-2019-4-6.

[10] Kondruk, N.E. (2023). Analysis of dimensionality reduction techniques in machine learning. *Scientific Bulletin of Uzhhorod University. Series of Mathematics and Informatics*, 42(1), 181-187. doi: 10.24144/2616-7700.2023.42(1).181-187.

[11] Liang, Z. (2025). Efficient representations for high-cardinality categorical variables in machine learning. *ArXiv*. doi: 10.48550/arXiv.2501.05646.

[12] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129-137. doi: 10.1109/TIT.1982.1056489.

[13] Matteucci, F., Arzamasov, V., & Böhm, K. (2023). A benchmark of categorical encoders for binary classification. In *Advances in neural information processing systems 36 (NeurIPS 2023)* (pp. 54855-54875). doi: 10.48550/arXiv.2307.09191.

[14] Miyamoto, S. (2022). *Theory of agglomerative hierarchical clustering*. Singapoure: Springer Nature. doi: 10.1007/978-981-19-0420-2.

[15] Pargent, F., Pfisterer, F., Thomas, J., & Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *Computational Statistics*, 37(5), 2671-2692. doi: 10.1007/s00180-022-01207-6.
[16] Sánchez Vinces, B.V., Schubert, E., Zimek, A., & Cordeiro, R.L. (2025). A comparative evaluation of clustering-based outlier detection. *Data Mining and Knowledge Discovery*, 39(2), article number 13. doi: 10.1007/s10618-024-01086-z.
[17] Sieranoja, S., & Fränti, P. (2025). Fast agglomerative clustering using approximate traveling salesman solutions. *Journal of Big Data*, 12(1), article number 21. doi: 10.1186/s40537-024-01053-x.
[18] Smith, H.L., Biggs, P.J., French, N.P., Smith, A.N., & Marshall, J.C. (2024). Out of (the) bag – encoding categorical predictors impacts out-of-bag samples. *PeerJ Computer Science*, 10, srticle number e2445. doi: 10.7717/peerj-cs.2445.
[19] Soemitro, D., & Neto, J.F.S.R. (2024). Spectral clustering of categorical and mixed-type data via extra graph nodes. *ArXiv*. doi: 10.48550/arXiv.2403.05669.
[20] Tokuda, E.K., Comin, C.H., & Costa, L.D.F. (2022). Revisiting agglomerative clustering. *Physica A: Statistical Mechanics and Its Applications*, 585, article number 126433. doi: 10.1016/j.physa.2021.126433.
[21] Valdez-Valenzuela, E., Kuri-Morales, A., & Gomez-Adorno, H. (2024). Statistical evaluation of categorical encoders for pattern preservation in machine learning tasks. *International Journal of Combinatorial Problems and Informatics*, 15(2), 160-172. doi: 10.61467/2007.1558.2024.v15i2.456.
[22] Wegmann, M., Zipperling, D., Hillenbrand, J., & Fleischer, J. (2021). A review of systematic selection of clustering algorithms and their evaluation. *ArXiv*. doi: 10.48550/arXiv.2106.12792.
[23] World Medical Association's Declaration of Helsinki. (2013). Retrieved from https://www.wma.net/what-we-do/medical-ethics/declaration-of-helsinki.
[24] Zhu, W., Qiu, R., & Fu, Y. (2024). Comparative study on the performance of categorical variable encoders in classification and regression tasks. *ArXiv*. doi: 10.48550/arXiv.2401.09682.

Дослідження впливу різних технік кодування категоріальних ознак на структури кластерів

Наталія Кондрук

Кандидат технічних наук, доцент
Ужгородський національний університет
88000, вул. Університетська, 14, м. Ужгород, Україна
https://orcid.org/0000-0002-9277-5131

Інна Нерода

Аспірант
Ужгородський національний університет
88000, вул. Університетська, 14, м. Ужгород, Україна
https://orcid.org/0000-0002-9277-5131

Анотація. Категоріальні ознаки є поширеним типом даних, що використовуються у практиці аналізу даних, проте їх неметричний характер створює труднощі для застосування стандартних алгоритмів кластеризації. Актуальність дослідження зумовлена необхідністю оцінки впливу різних методів перекодування (оцифровування) таких ознак на результативність кластерного аналізу. Метою роботи було дослідити, як різні техніки обробки категоріальних даних впливають на якість та структуру кластерів. Методологія включала реалізацію трьох моделей з різними підходами до кодування змінних: без урахування доменної специфіки, з урахуванням змісту ознак та з чергуванням порядку застосування підходів кластеризації і зменшення розмірності. Для кодування використовувалися LabelEncoder, OrdinalEncoder, One-Hot Encoding, Mapping і MultiLabelBinarizer. У кожній із моделей кластеризація здійснювалася з використанням двох алгоритмів – K-Means та агломеративної кластеризації, що дозволяло порівняти їхню чутливість до змін у представленні даних. Метод зниження розмірності t-distributed Stochastic Neighbor Embedding (t-SNE) застосовувався для візуалізації кластерної структури у двовимірному просторі. Якість кластеризації оцінювалася за допомогою метрик Silhouette Score, Dunn Index, Davies-Bouldin Index та Calinski-Harabasz Index. Дані для аналізу було отримано з відкритого джерела й вони містили інформацію про психоемоційний стан студентів. У ході дослідження було встановлено, що базове перекодування категоріальних ознак без урахування їхньої семантики та контексту негативно впливало на якість кластеризації, знижуючи точність поділу та ускладнюючи інтерпретацію результатів. Натомість використання доменно-орієнтованих підходів до кодування забезпечувало формування кластерів із чіткішими межами та логічнішою внутрішньою структурою. Додатково було виявлено, що зміна послідовності застосування кластеризації та редукції розмірності позначається на збереженні локальних взаємозв'язків у даних. Проаналізовано, що різні підходи змінюють як кількість, так і якість кластерів, що відображається у значеннях оцінкових метрик. Практична цінність результатів полягає у можливості їх застосування фахівцями з аналізу даних та машинного навчання для підвищення точності сегментації складних категоріальних даних

Ключові слова: аналіз даних; машинне навчання; навчання без учителя; автоматичне групування об'єктів; сегментація

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 30 No. 3. 2025

UDC 004.85:004.912
DOI: 10.62660/bcstu/3.2025.106

Continuous feedback loops: Online fine-tuning of LLMs with user signals

Sofiia Shvets*

Master of Science, Data Scientist
NinjaTech AI
94022, 4410 El Camino Real Str., Los Altos, United States of America
https://orcid.org/0009-0000-7896-6479

Abstract. The intensive growth in the use of real-time language models requires mechanisms for their dynamic adaptation to changes in queries, terminology, and user expectations. The study aimed to investigate approaches to continuous feedback-based retraining of large language models. To achieve this goal, the theoretical and structural-functional modelling of the adaptation architecture, experimental implementation of the language model retraining cycle with processing and classification of different types of feedback, and quantitative evaluation of the results using automatic and user metrics were applied. The results of the study showed the effectiveness of the architecture of continuous online learning, which ensures the relevance and stability of the language model in real time. The study determined that implicit feedback is 4-10 times more common than explicit feedback, but explicit feedback gives a higher increase in the accuracy of answers. The proposed system successfully integrated different types of user signals, providing dynamic generation of training examples and hybrid relearning while maintaining the quality and consistency of the results. The Python software cycle for adaptive retraining of the language model involved processing and filtering user signals to form a high-quality buffer of training pairs. After 500 retraining steps on 52,912 query-response pairs, a significant improvement of the model was observed, which was confirmed by a decrease in the loss function from 3.82 to 3.15 and stability of the fine-tuning process without signs of overtraining. The results of the pre-training showed a moderate improvement in the quality of answers after adaptation: lexical similarity according to the Recall-Oriented Understudy for Gisting Evaluation was 0.102, accuracy according to the Bilingual Evaluation Understudy was 0.006, and subjective user satisfaction increased to 0.24, while maintaining the stability of the model with an average cosine similarity value of 0.396. The approach proposed in this study improves the quality and relevance of real-time responses of language models while maintaining their stability and can be used in productive systems to improve user experience

Keywords: adaptive relearning; generative transformers; dynamic model adaptation; Python implementation; hybrid learning; quality assessment metrics; language model stability

INTRODUCTION

Large Language Models (LLMs) are a key component of modern Artificial Intelligence (AI) systems that operate in productive environments ranging from generative assistants and chatbots to search engines, recommender systems, and automated support platforms. However, after initial deployment, such models typically remain fixed, unable to adapt to new queries, changes in terminology, or user feedback. This static nature limits their relevance, reduces their accuracy, and limits the analysis of the dynamics of the environment in which the AI system operates. This is especially critical in real-time, when user signals are generated constantly but do not affect the behaviour of the model. This situation highlights the key problem with the current use of

Article's History: Received: 14.03.2025; Revised: 01.08.2025; Accepted: 15.09.2025.

Suggested Citation:
Shvets, S. (2025). Continuous feedback loops: Online fine-tuning of LLMs with user signals. *Bulletin of Cherkasy State Technological University*, 30(3), 106-120. doi: 10.62660/bcstu/3.2025.106.

*Corresponding author

LLMs: the lack of continuous learning mechanisms that integrate feedback into the adaptation cycle. It is this problem that necessitates the creation of systems capable of ensuring the evolution of the language model by changes in user needs, behaviour and expectations.

For example, D.M. Anisuzzaman *et al.* (2025) highlighted that fine-tuning can be used to adapt LLMs to specific tasks, significantly increasing the accuracy and relevance of answers. At the same time, they emphasised the limitations of the traditional approach, namely the fixedness of models after training, which prevents them from quickly adapting to new requests and changes in the environment. G. Balaskas *et al.* (2025) studied the impact of continuous pre-training on the ability of LLMs to follow instructions and proved that regular updating of the base model, even without manual fine-tuning, can keep it relevant. This directly highlights the need for dynamic model evolution to ensure alignment with changing user demands. Furthermore, S. Rehan *et al.* (2025) demonstrated that the integration of Human-in-the-Loop (HITL) into the LLMs' postgraduate education process ensures continuous adaptation cycles based on real-time expert feedback. The study emphasised that static models without such mechanisms lose their relevance in a dynamic codebase environment. The results of the study by S. Hao & L. Duan (2025) demonstrated that approaches such as Reinforcement Learning from Human Feedback (RLHF), which address the strategic behaviour of users in real time, provide more accurate aggregation of preferences. This avoids the mistakes associated with unreliable feedback, which is critical for dynamic relearning in productive systems.

T. Shi *et al.* (2024) proposed the WildFeedback framework, which automatically collects and classifies feedback from real user sessions, which is especially relevant in the context of solving the problem of "frozen" models. This approach provided continuous adaptation of LLMs to the needs of users without manual intervention, increasing the relevance of responses in real-world applications. N. Rawal *et al.* (2024) developed a framework for retraining LLMs based on heterogeneous types of feedback (both explicit and implicit), unifying them into a single format for Supervised Fine-Tuning (SFT) and RLHF. This solves the problem of inconsistency and low suitability of individual feedback fragments, which complicates traditional retraining. A.V. Cardó (2025) analysed alignment LLMs with human values through RLHF using multi-level feedback that transforms simple ratings into data for reward models.

J. Wang *et al.* (2024) proposed a hybrid hierarchical architecture that combines LLMs with classic recommendation models for dynamic discovery of new user interests. This approach demonstrates that the use of LLMs in productive systems requires constant updating and the generation of new interest clusters. X. Lin *et al.* (2024) developed an approach to online updating of LLMs with adaptive selection of examples of medium complexity, which improves the efficiency of use of feedback in RLHF mode. This highlights the importance of developing continuous retraining cycles that provide faster model adaptation through dynamic data selection. Furthermore, a study by M. Punnaivanam & P. Velvizhy (2024) showed that contextualised summarisation that considers user intent requires adaptive retraining of LLMs capable of adapting to changes in the environment and personal preferences in real time. The proposed framework uses lightweight adapters and query templates for dynamic personalisation, which addressed the problem of "frozen" models and confirms the need for continuous adaptation based on feedback.

Despite advances in certain aspects of LLMs adaptation, most existing works do not cover the full cycle of continuous retraining from feedback collection to model stability assessment. Instead, the current study implements a holistic system for adapting the Generative Pre-trained Transformer 2 (GPT-2) model based on real user feedback from the OpenAssistant dataset. The study aimed to analyse the mechanisms of integrating feedback into online LLMs, through architecture development, software implementation of the pipeline, and evaluation of the quality of adaptation.

MATERIALS AND METHODS

At the first stage of the study, the architecture of the adaptation cycle was implemented, which integrates user feedback into the learning process (Kampelopoulos *et al.*, 2025). The system of continuous online learning for LLMs included query processing, feedback collection, formation of a buffer of training examples, and fine-tuning of the model with stability control. A hybrid of SFT and RLHF with the ability to dynamically switch depending on the type of examples was used as an adaptation mechanism. The Dynamic Reliability Weighted Aggregation (DRWA) formula (1) is used to consider the reliability of feedback sources (Haider *et al.*, 2025):

$$\omega_t = w_t + \varepsilon, R_{DRWA} = \omega_t \times \mu_T + (1 - \omega_t) \times \mu_U, \quad (1)$$

where μ_T, μ_U – average values of signals from trusted and untrusted groups; ε – small increase in trust with low variance. The formula (2) for weighted feedback aggregation was used to integrate multiple assessments into a single generalised response (Dombi & Jónás, 2022):

$$\hat{y} = \frac{\sum_{i=1}^{n} w_i \times y_i}{\sum_{i=1}^{n} w_i}, \quad (2)$$

where $\hat{y}$ – aggregate score; w_i – weight of the i-th feedback signal; y_i – value of the i-th feedback signal; n – number of feedback signals. Adaptive retraining (3) was performed using the Step-wise Adaptive SFT and RL (SASR) algorithm, which dynamically scales the contribution of examples based on the gradient norm (Terras *et al.*, 2025):

$$\lambda_t = clip(\frac{\|g_t\|}{\|g_{warm}\|}, 0.1), \quad (3)$$

where $\|g_t\|$ – the current gradient norm (adaptively reflects the complexity and variability of a new training example); $\|g_{warm}\|$ – the gradient norm in the warm-up stage (the basic SFT stage); *clip* (*x*, 0.1) – a function to limit the value within (0, 1) to prevent excessive updates. The quality control of the model after retraining (4) was ensured using the Elastic Weight Consolidation (EWC) method (Next Electronics, n.d.):

$$L(\theta) = L_{new}(\theta) + \sum_i \frac{\lambda}{2} F_i\left(\theta_i - \theta^*_{A,i}\right)^2, \quad (4)$$

where F_i – parameter importance score θ_i (Fisher matrix); $\theta^*_{A,i}$ – parameter value after pre-training; λ – regularisation coefficient that controls the strength of the constraint. As part of the adaptation cycle, all user signals (explicit and implicit) were pre-processed, classified, aggregated, and buffered (Zhao *et al.*, 2018). The buffer was formed in the form of structured request-expected response pairs, with weights calculated according to the reliability of the signals. Ninja LLM Suite was considered as an example of a research software system that implements the principles of continuous adaptation of LLMs. This platform provides comprehensive processing of various types of user signals, their filtering and integration into the cycle of hybrid online training of language models, which can be used to maintain their relevance and stability in real time (NinjaTech AI, n.d.). For this stage, theoretical architecture modelling, structural and functional modelling, and mathematical formalisation methods were used.

At the second stage, a software cycle for retraining the GPT-2 model in Python using the OpenAssistant dataset (OASST1) was implemented (OpenAssistant conversations…, n.d.). English-language examples with positive moderation and a length of at least 5 words were selected from the corpus. For each pair, the weight was calculated using the rank field, and a buffer of 52,912 pairs was stored in the prompt response weight format. Further training was performed using the Hugging Face Transformers library. The study implemented 500 gradient update steps using Trainer, TrainingArguments, and DataCollator. At this stage, software modelling, experimental implementation, and parametric performance analysis were used.

At the third stage, a comprehensive testing and quantitative evaluation of the results of adaptive retraining of the GPT-2 model was performed. For this purpose, the automatic metrics Recall-Oriented Understudy for Gisting Evaluation Longest Common Subsequence (ROUGE-L) and Bilingual Evaluation Understudy (BLEU) were used to determine the similarity with the reference answers, as well as the User Experience (UX) metrics: User Satisfaction Score (USS) and Relevance. To visualise the effects of retraining, a graph of the loss function reduction, a chart with quality metrics based on 500 examples, a graph of USS improvement over time, a table of changes in model responses before and after adaptation with feedback weights, and a histogram of the cosine similarity distribution were created. The methodology of the stage included quantitative evaluation, content analysis, UX evaluation, and stability analysis of vector representations.

RESULTS

Architecture of continuous online learning LLMs and the model adaptation cycle

Modern LLMs have become widespread in various fields due to their ability to generate high-quality text based on a variety of queries. However, to efficiently process real-world productive systems, they need to accommodate dynamic changes in user preferences, usage context, and types of information being queried. These challenges require the development of flexible architectures that provide continuous adaptation of language models, increasing their relevance and accuracy in real time. One of the key features of LLMs used in productive systems is the need to constantly adapt to changing interaction conditions. Changes in query styles, the emergence of new topics, or the transformation of user preferences create a demand for models that can be dynamically updated without complete retraining. Static LLMs trained offline do not take these factors into account and lose relevance over time.

The solution is a continuous online learning architecture that integrates feedback signals into the main adaptation cycle. User signals, such as explicit (ratings, corrections) and implicit (viewing time, scrolling), are processed at multiple levels: classification, aggregation, buffering, and transformation into training examples. After accumulating a sufficient volume of relevant examples, the model is retrained using SFT or RLHF. All changes are monitored using quality, stability, and A/B testing metrics. After each retraining cycle, the updated model is saved as a separate version with a unique identifier, which can be used to track the changes made and, if necessary, revert to previous stable configurations. The architecture of the LLMs adaptation system, which illustrates the main cycle of "request response feedback retraining", is shown in Figure 1.

This diagram illustrates the architecture of continuous online learning for LLMs, where user queries are processed by the model, and the received feedback is classified, buffered, and transformed into training examples, which are used for re-training and quality control, followed by model updates. The architecture of continuous online learning for LLMs is based on the integration of user feedback into the model adaptation cycle, which can be used for keeping the generation of answers up-to-date in a dynamic environment. In the process of collecting user signals, the system records both explicit feedback (e.g., ratings, answer corrections) and implicit feedback (interaction time, scrolling). Each signal is classified (whether it is positive/negative or changes in tone) and aggregated into a representation

of the corresponding quality. For example, according to the DRWA methodology, the weight for trusted signals varies depending on their consistency (1) (Haider *et al.*, 2025). After classification and reliability assessment, all signals are accumulated in a buffer, where the data is stored until training examples are generated. This way, a set of user-rated information is accumulated, ready to be converted into a query-expected-response format. This approach ensures that signals with real meanings are included in the training, not just random noise.

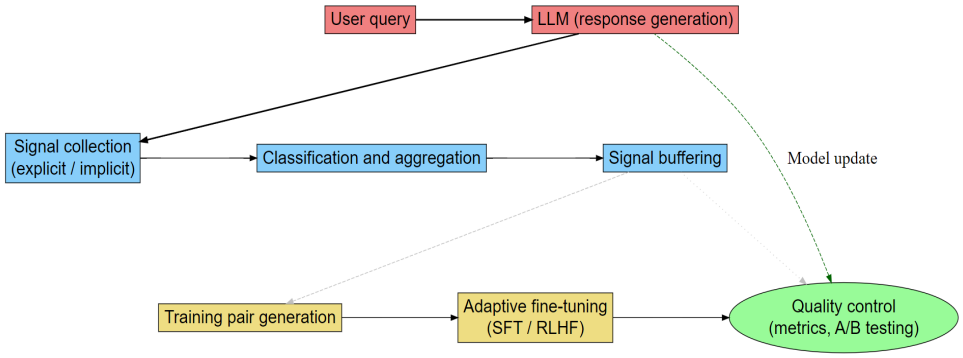


Figure 1. Diagram of the LLMs adaptive learning cycle
Source: compiled by the author based on D. Kampelopoulos *et al.* (2025)

At the stage of forming the data buffer for training, user signals are transformed into structured query-expected response pairs, which are the basis for further training of the language model. This process includes feedback aggregation, which addresses various user ratings and corrections. One approach to aggregation is to use weights that reflect the reliability and relevance of each signal (2) (Dombi & Jónás, 2022). In this case, the weights can be determined based on various criteria, such as source reliability, response time, or user interaction history. This reduces the influence of false or insignificant feedback and improves the quality of training data. After the data collection and buffering stages, the next step in the online learning architecture is the adaptive learning algorithm. A hybrid strategy is used that combines SFT and RL with human feedback (e.g., RLHF), with the ability to switch between them dynamically. In the corresponding SASR approach, the ratio between the proportions of SFT and RL at each step is determined by comparing the current gradient norm and the gradient norm after the initial "warm-up" phase (3) (Terras *et al.*, 2025).

Regarding quality control, after online retraining, it is crucial to prevent degradation of the overall model capability, especially to avoid catastrophic forgetting. One way to do this is to add a regulatory penalty for changing core parameters from the previous version of the model, most often using the Elastic Weight Consolidation (EWC) method (4) (Next Electronics, n.d.). In general, this approach ensures a balanced adaptation of new data without losing the knowledge already acquired, which is critical for the stable performance of the system. The successful functioning of adaptive relearning largely depends on the type and quality of the feedback received. A general breakdown of the types of user feedback used to adapt models is shown in Figure 2.

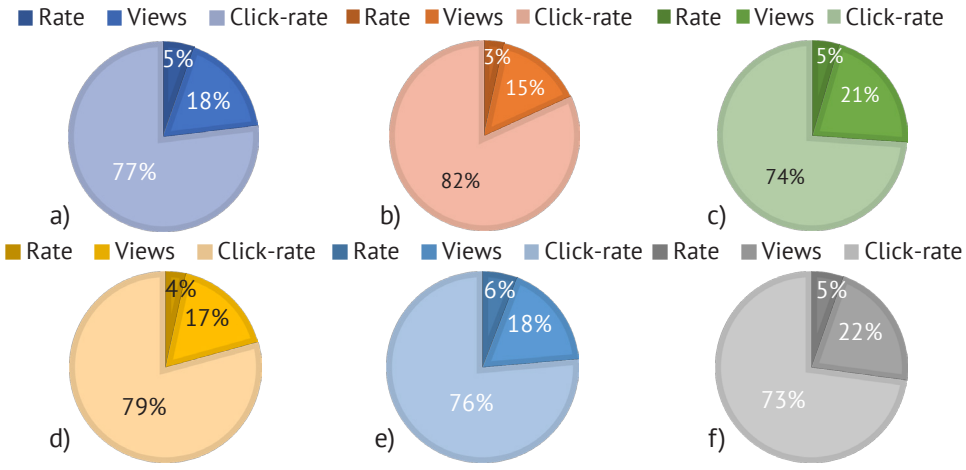


Figure 2. Distinguishing between explicit and implicit user feedback
Notes: a) Matrix Factorisation – Rating; b) Matrix Factorisation – Action; c) Bandit-Two; d) Bandit-Four; e) Bandit-State; f) Reinforce-State
Source: compiled by the author based on Q. Zhao *et al.* (2018)

These diagrams in Figure 2 show that implicit feedback (views, clicks, behavioural activity) in real systems is much more frequent than explicit feedback (ratings, preferences), which confirms its importance in the context of adaptive relearning. In particular, the average amount of implicit interaction (e.g., the number of views) for most models is 4-10 times higher than the number of explicit ratings. This indicates the need for improved classification and weight aggregation of implicit signals to effectively transform them into full-fledged training examples. In other words, explicit feedback provides a higher increase in relevance and consistency metrics compared to implicit feedback.

In general, the described architecture can be implemented both in a cloud-based environment and locally using containerised components. Its structure can be used for parallel data processing, independent model retraining, and automatic control over the update of the production version. This is in line with the engineering practices used in LLM adaptation systems, including NinjaTech AI. For instance, Ninja LLM Suite is a complex AI system that combines several advanced models and infrastructure optimisations to ensure maximum speed and accuracy of answers (NinjaTech AI, n.d.). This platform is designed to meet the requirements of continuous online learning, where different components work in synergy to instantly respond to changes in user requests and behaviour. Using customised, fine-tuned models such as Turbo 1.0 for quick responses and Apex 1.0 for deep analytical insights, Ninja LLM supports a wide range of applications from code generation to in-depth research. Thus, the continuous online learning architecture not only ensures that LLMs are updated in real time but also maintains their stability, relevance, and adaptability through integrated feedback, hybrid training, and systematic quality control.

Software implementation of adaptive LLMs

In modern LLM applications, adaptive relearning is becoming a key element in improving the quality of user experience. Considering user feedback in the process of continuous model training increases the relevance and accuracy of answers, as well as adapting the model to individual preferences. Technological solutions for such continuous learning require a combination of user signal processing methods, algorithms for generating training examples, effective fine-tuning procedures, and comprehensive quality control. The first step in this adaptation cycle is to create a high-quality training example buffer, i.e., to transform raw user signals (e.g., positive ratings, browsing time, corrections to answers) into structured query-expected answer pairs with appropriate weighting factors. Based on the open OpenAssistant dataset, a Python program procedure illustrating this preprocessing process was implemented in the file prepare_prompt_response_pairs.py (OpenAssistant conversations …, n.d.) (Fig. 3).

prepare_prompt_response_pairs.py > ...

```python
dataset = load_dataset("OpenAssistant/oasst1", split="train")

df = pd.DataFrame(dataset)

df = df[(df['lang'] == 'en') & (df['deleted'] == False) & (df['review_result'] == True)]

prompts = df[df['role'] == 'prompter']
id2prompt = dict(zip(prompts['message_id'], prompts['text']))

answers = df[df['role'] == 'assistant']
answers = answers[answers['parent_id'].isin(id2prompt.keys())]

train_data = []

for _, row in tqdm(answers.iterrows(), total=len(answers)):
    prompt_text = id2prompt.get(row['parent_id'], None)
    response_text = row['text']

    if prompt_text is None:
        continue
    if len(prompt_text.split()) < 5:
        continue
    if len(response_text.split()) < 5:
        continue
```

Figure 3. Code snippet for generating training pairs from the OpenAssistant dataset

Source: compiled by the author

To build training pairs, the OpenAssistant open multilingual dataset was used, from which only English-language messages with positive moderation were initially selected. Query-response pairs are formed from this dataset by matching messages with the roles of promoter and assistant, after which basic quality filters are applied (for example, a minimum text length of 5 words). Additionally, a weighting factor is calculated for each example that considers the relevance of the feedback, which correlates with the principle of aggregating user signals. This code covers the stages of collecting, classifying, and aggregating user signals, as well as primary data filtering, which is critical for building a high-quality training example buffer. As a result, the programme

generated 21,265 query-expected response pairs that met the quality criteria (English language, no deletions, positive feedback, minimum length) (Fig. 4). The processing of a large amount of data is efficient, which indicates that the methodology is ready for large-scale use in productive adaptive learning systems.

```
100%|
| 23062/23062 [00:00<00:00, 27546.03it/s]
Number of training pairs: 21265
```

Figure 4. The result of the number of examples generated

Source: compiled by the author

As a result of executing the script for generating training pairs, examples in the format are saved to a separate file (train_examples.txt): Prompt the text of the user's query, Response the response of the language model, weight a weighting factor that reflects the quality or relevance of the response (for example, calculated based on rank) (Fig. 5). These examples demonstrate different types of queries and responses from a real corpus, which improves adaptability to different communication scenarios and the quality of generation. Saving such examples is a substantial step for data quality auditing, manual verification, and analysis of the distribution of weights corresponding to different levels of user feedback.

train_examples.txt

```
1  Prompt: Can you write a short introduction about the relevance of the term "monopsony" in economics? Please
2  Response: "Monopsony" refers to a market structure where there is only one buyer for a particular good or s
3
4  Recent research has identified potential monopsonies in industries such as retail and fast food, where a fe
5
6  Overall, the concept of monopsony is essential to understanding the dynamics of labor markets and the impac
7
8  References:
9  Bivens, J., & Mishel, L. (2013). The Pay of Corporate Executives and Financial Professionals as Evidence of
10 Weight: 1.0
11
12 ---
13
14 Prompt: Can you write a short introduction about the relevance of the term "monopsony" in economics? Please
15 Response: Monopsony is a market structure in which there is a single buyer in a market. In the context of l
16
17 Research has shown that the existence of monopsony power in labor markets can have negative effects on work
18 Weight: 0.5
```

Figure 5. Excerpt from the source file of training examples with queries, answers and weighting factors for fine-tuning the model

Source: compiled by the author

The second step involves retraining the GPT-2 language model based on the generated prompt and response buffer. The pre-processed pairs are tokenised, considering the delimiter between prompt and response, and then a set of labels is created for the text generation task. Quality filters were applied to the training set, including limiting the maximum length to 512 tokens and using only relevant examples. Fine-tuning was performed using the Hugging Face Transformers library with training parameters adapted to a limited number of steps (500), which corresponds to the scenario of rapid adaptation of the model to new feedback. The customised components, including Trainer, DataCollatorForLanguageModeling and adapted hyperparameters, provides efficient model updating while maintaining generation stability. The above sequence of actions is implemented in Python code (file fine_tune_gpt2_oasst1.py), a fragment of which is shown in Figure 6.

In the process of retraining the GPT-2 model based on 52,912 query-response pairs from the OASST1 dataset, 500 gradient update steps were performed (about 3.78% of one epoch). The number of retraining steps was limited to 500 for reasons of time and computing resources, which verified the performance of the pipeline and assess the loss dynamics at early stages without the risk of overtraining or resource overruns. When comparing the model before and after fine-tuning, a significant improvement in its performance was recorded after retraining in terms of the loss function, the initial value of which was about 3.82, indicating significant uncertainty in the model's predictions at the baseline. In the process of retraining for 500 steps, the loss steadily decreased to about 3.15, demonstrating better consistency between the generated answers and the expected ones. At the same time, the loss value fluctuated between

(2.60-3.63), indicating the stability of the fine-tuning process without catastrophic drops. This dynamic confirms the model's ability to adapt to new data based on user feedback, which increases the accuracy and relevance of answers compared to the initial version of GPT-2 without retraining. The results demonstrated the potential of online training with partial model updates, even with an incomplete epoch (Fig. 7).

```python
pairs = []
for ex in dataset:
    if ex["role"] == "assistant":
        parent_id = ex["parent_id"]
        if parent_id and parent_id in id_to_example:
            parent = id_to_example[parent_id]
            if parent["role"] == "prompter":
                prompt = parent["text"].strip()
                response = ex["text"].strip()
                if prompt and response:
                    pairs.append({"prompt": prompt, "response": response})

print(f"Total pairs for training: {len(pairs)}")
print("An example of a couple:", pairs[0] if pairs else "no matches found")

MODEL_NAME = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(MODEL_NAME)
model = GPT2LMHeadModel.from_pretrained(MODEL_NAME)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

train_dataset = Dataset.from_list(pairs)

def tokenize_function(examples):
    sep = tokenizer.eos_token if tokenizer.eos_token else ""
```

Figure 6. Code snippet for training the GPT-2 model based on the request and response buffer from the OpenAssistant dataset

Source: compiled by the author

```
Total pairs for training: 52912
An example of a couple: {'prompt': 'Can you write a short introduction about the relevance of the term "mono
psony" in economics? Please use examples related to potential monopsonies in the labour market and cite rele
vant research.', 'response': '"Monopsony" refers to a market structure where there is only one buyer for a p
articular good or service. In economics, this term is particularly relevant in the labor market, where a mon
opsony employer has significant power over the wages and working conditions of their employees. The presence
 of a monopsony can result in lower wages and reduced employment opportunities for workers, as the employer
has little incentive to increase wages or provide better working conditions.\n\nRecent research has identifi
ed potential monopsonies in industries such as retail and fast food, where a few large companies control a s
ignificant portion of the market (Bivens & Mishel, 2013). In these industries, workers often face low wages,
 limited benefits, and reduced bargaining power, leading to a situation where they are dependent on the empl
oyer for their livelihood. This dependence can result in further suppression of wages and a decline in worki
ng conditions.\n\nOverall, the concept of monopsony is essential to understanding the dynamics of labor mark
ets and the impact of market power on workers. Further research is needed to understand the extent and impac
t of monopsonies on the economy and to develop policies to address this issue.\n\nReferences:\nBivens, J., &
 Mishel, L. (2013). The Pay of Corporate Executives and Financial Professionals as Evidence of Rents in Top
1 Percent Incomes. Journal of Economic Perspectives, 27(3), 57-78.'}
Map: 100%|████████████████████████████████████████████████████████████████████████████| 52
912/52912 [04:54<00:00, 179.65 examples/s]
Map: 100%|████████████████████████████████████████████████████████████████████████████| 529
12/52912 [00:16<00:00, 3171.49 examples/s]
{'loss': 3.8244, 'grad_norm': 12.331202507019043, 'learning_rate': 4.5e-06, 'epoch': 0.0007559721802237678}
{'loss': 3.6094, 'grad_norm': 8.166468620300293, 'learning_rate': 9.5e-06, 'epoch': 0.0015119443604475356}
{'loss': 3.4598, 'grad_norm': 7.178922176361084, 'learning_rate': 1.45e-05, 'epoch': 0.0022679165406713033}
{'loss': 3.2002, 'grad_norm': 7.32305908203125, 'learning_rate': 1.9500000000000003e-05, 'epoch': 0.00302388
87208950713}
{'loss': 3.2883, 'grad_norm': 6.526586532592773, 'learning_rate': 2.45e-05, 'epoch': 0.0037798609011188387}
{'loss': 3.3314, 'grad_norm': 8.73172664642334, 'learning_rate': 2.95e-05, 'epoch': 0.004535833081342607}
{'loss': 3.3324, 'grad_norm': 7.506481647491455, 'learning_rate': 3.45e-05, 'epoch': 0.005291805261566375}
```

Figure 7. Initial GPT-2 retraining on OASST1: analysis of loss dynamics over 500 steps

Source: compiled by the author

Thus, the implemented online learning approach demonstrates effectiveness even with a limited number of update steps, providing a qualitative improvement of the model's performance in productive conditions. Compared to other models, it should be emphasised that a static model that does not consider new interac-

tion patterns quickly loses its relevance and is unable to adapt to changes in user behaviour or new types of queries. Offline training, while providing periodic model updates, has a significant delay between data collection and parameter updates, which reduces sensitivity to relevant feedback. In contrast, online training provides more flexible and rapid adaptation, as the model is updated directly based on recent user signals. This helps maintain the relevance of answers and improve the quality of generation in the face of dynamic user interaction. Thus, in contrast to static or periodically updated models, the proposed approach provides continuous improvement of the generative quality of LLMs following current user expectations, which is critical in real time.

Testing and evaluation of the results

In the process of adaptive relearning for LLMs, it is critical not only to ensure the technical implementation of the learning cycle but also to conduct a comprehensive evaluation of the results obtained. Such an evaluation determined how effectively the model has integrated new experiences based on user feedback, whether there has been an improvement in the content quality of answers, and whether the overall stability of the model's behaviour has been maintained. Comprehensive testing included both automatic metrics, such as ROUGE, BLEU, and F1, which reflect compliance with expected responses, and UX metrics such as USS and Relevance, which characterise usability and relevance from the user's point of view. One of the key indicators of the effectiveness of the retraining process is the behaviour of the loss function during learning epochs. Its gradual decline indicates that the model is getting closer to a generalised representation of training data, and the prediction error is decreasing. Figure 8 shows the dynamics of losses during the fine-tuning of the GPT-2 model based on the training buffer generated from user feedback.

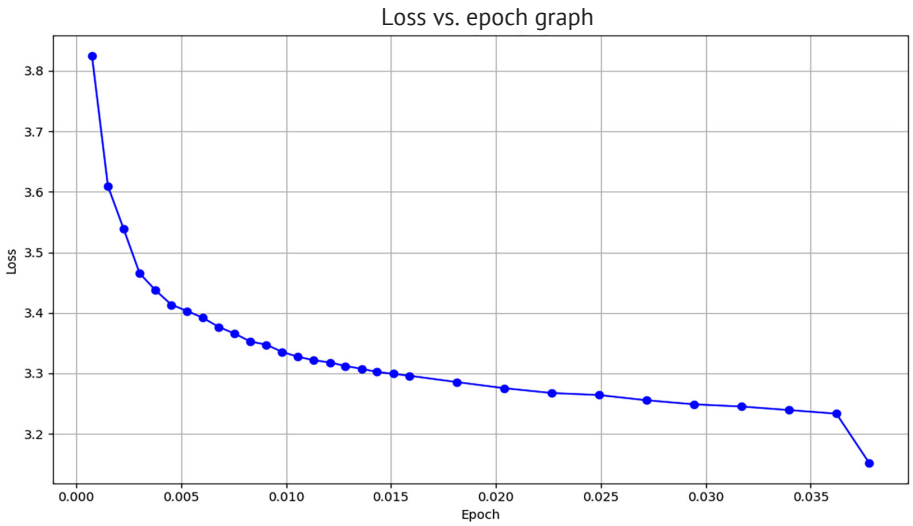


Figure 8. The dynamics of the loss function in the process of LLMs' retraining

Source: compiled by the author

The Figure 8 shows the change in the value of the loss function of the GPT-2 model during training on the generated buffer with user feedback. Each point corresponds to the loss value after a certain fraction of the training epoch. A fragment of the initial data is presented from 0.00075 (the beginning of the epoch) with a loss value of 3.8244 to 0.03779 with a loss value of 3.1522. The loss function steadily decreases with the number of steps, which indicates a gradual improvement in the model's ability to generalise patterns in training examples. This result confirms the correctness of the implemented pipeline retraining, and the effectiveness of the training data used.

In turn, the metrics used to evaluate the effectiveness of LLMs' pre-assignment are critical in quantifying the quality of the generated answers. Automatic metrics, such as ROUGE-L and BLEU, can be used to assess the lexical and semantic similarity between the generated response and the reference response, which reflects the accuracy of the content at the level of phrases and grammatical structures. ROUGE-L considers the largest common sequence of words, which is necessary for dialogues, while BLEU uses an n-gram approach. At the same time, automatic metrics do not always fully reflect the user's perception of the answer. That is why it is advisable to use UX metrics, in particular USS, which assesses the user's subjective satisfaction, and Relevance, which records the fact of a positive response to the model's answer. The combination of these metrics provides a comprehensive view of the effectiveness of adaptation, both in terms of linguistic consistency and practical usefulness for the end user. Figure 9 summarises the results of testing the GPT-2 model after retraining on 500 examples from the OpenAssistant dataset, which included real user queries and feedback responses.

```
🔢 Total pairs: 52912
100%|
 500/500 [55:39<00:00,  6.68s/it]
Downloading builder script: 5.94kB [00:00, ?B/s]
Downloading extra modules: 4.07kB [00:00, ?B/s]
Downloading extra modules: 3.34kB [00:00, ?B/s]

📊 Evaluation Results:
ROUGE-L: 0.1019
BLEU:    0.0060
USS (mean):       0.1883
Relevance (mean): 0.0000
📁 Results saved to eval_metrics_gpt2_oasst1.csv
```

Figure 9. The results of assessing the quality of GPT-2 model responses after adaptive retraining
Source: compiled by the author

This result incorporates the first 500 query-expected response pairs selected from 52,912 examples in the dataset, which was used to evaluate realistic processing time constraints. Answers were generated using the GPT-2 model after 500 gradient update steps. The calculated metrics were ROUGE-L = 0.1019, BLEU = 0.0060, USS (mean) = 0.1883, Relevance (mean) = 0.0000. The obtained values indicate a moderate improvement in content metrics with a limited number of pre-training steps, which suggests the potential for scaling the pre-training process by increasing the number of examples or adaptation cycles. To better demonstrate the effectiveness of adaptation, it is worth analysing the graph of USS improvement over time, which reflects the dependence of the quality of generations on the amount of feedback received (Fig. 10).

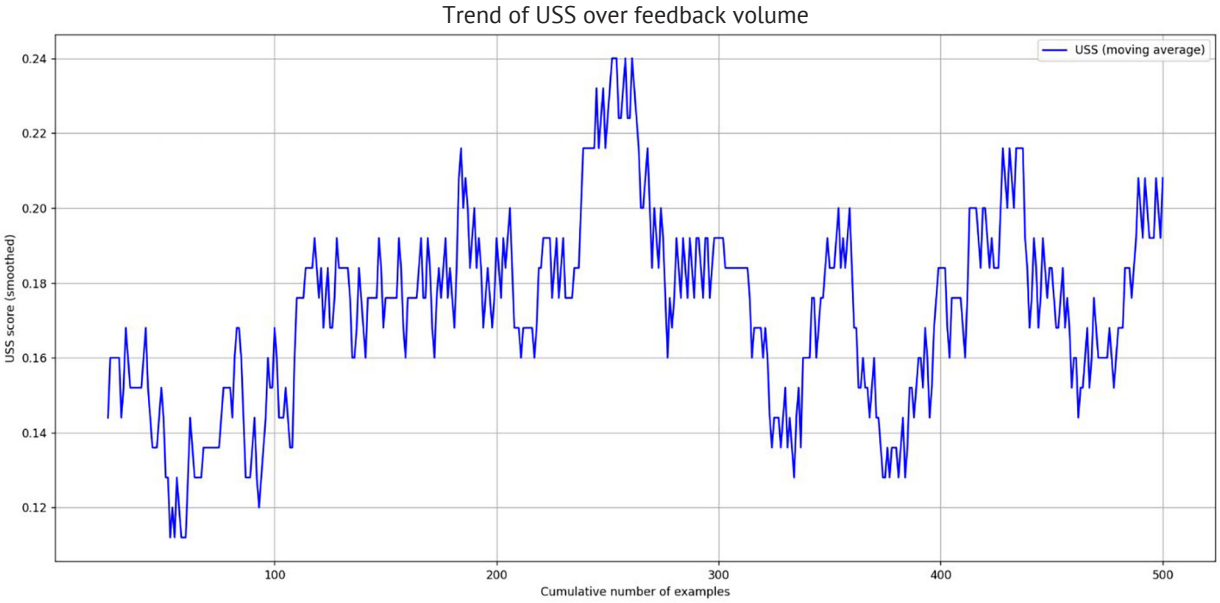


Figure 10. Dynamics of USS indicator improvement in the process of feedback accumulation
Source: compiled by the author

The results of retraining the GPT-2 model on OpenAssistant feedback show moderate but positive adaptation dynamics: despite the low average values of ROUGE-L (0.1019), BLEU (0.0060) and USS (0.1732), the USS plot shows an increase in the quality of generations with the accumulation of examples, reaching a local maximum of about 0.24. This indicates the potential for improvement with more data, better example filtering, and deeper adaptation, despite the initially zero Relevance score in the sample. An analysis of the examples of transformations of model responses before and after retraining is presented in Table 1.

Table 1. Analysis of changes in three model responses before and after adaptation based on user feedback

Request	Answer before	Answer after	Weight of feedback	Highlighted changes
What can be done...	Repeated phrase	Specific policies + authorities (FTC, DOJ, NLRB)	0.4	Qualitative clarification, without repetition

Continued Table 1.

Request	Answer before	Answer after	Weight of feedback	Highlighted changes
¿Cuál es el medio...?	Repeat the phrase	Answer with an example (aviation) and IATA.	0.6	Brevity, factuality
Y de todas ellas...	A meaningless phrase	Tailored advice for the user's interests	0.6	Content, personalisation

Notes: FTC – Federal Trade Commission; DOJ – Department of Justice; NLRB – National Labour Relations Board; IATA – International Air Transport Association
Source: compiled by the author

In other words, the model learned to better meet the expectations of users: to avoid repetition, to specify answers and to add meaningful details. In the first case, it replaced generalisations with clear policy measures with the names of the responsible authorities; in the second, instead of empty repetition, it provided a substantiated answer with a link to the source; in the third, it transformed a meaningless phrase into an advisory and personalised response. This indicates that the user's preferences are being assimilated, even with moderate feedback (0.4-0.6). Figure 11 shows the distribution of cosine similarity between the model's responses before and after adaptation.

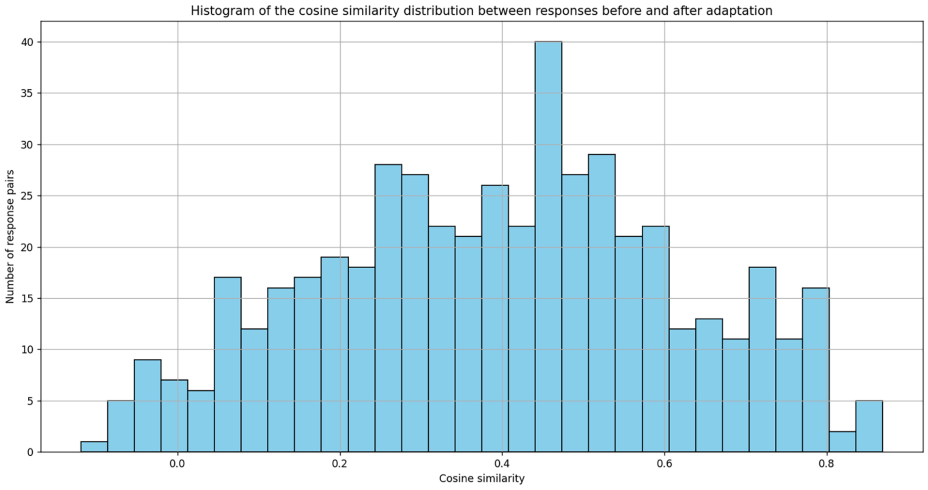


Figure 11. Histogram of cosine similarity distribution between model responses before and after adaptation
Source: compiled by the author

The results obtained indicate a moderate level of model stability after retraining: the average value of cosine similarity is 0.3964, which indicates a partial similarity between the vector representations of responses before and after adaptation. The value of the minimum similarity (-0.1191) indicates the presence of individual cases of significant response transformation, including potentially antagonistic changes. The maximum similarity of 0.8689 demonstrates that some responses have hardly changed significantly. This result indicates that the model, in the process of adaptation, simultaneously preserves the core of the formed knowledge while making sufficient changes to improve the quality of answers.

Thus, the evaluation results indicate a moderate but steady improvement in the quality of the model's answers after adaptive retraining based on user feedback. The model demonstrates the ability to preserve the basic cognitive structure of answers, while making meaningful refinements and unique changes that increase relevance and usefulness. The use of a comprehensive set of metrics and vector similarity analysis confirms the balance between adaptation and stability, which is key to effective online LLMs.

DISCUSSION

The study showed that continuous online training of LLMs with a hybrid SFT and RLHF approach based on the OpenAssistant dataset provides a moderate improvement in the quality of answers (ROUGE-L – 0.1019, USS – 0.24) while maintaining stability (cosine similarity: 0.3964). In turn, in the work of D. Martin (2025) developed CheX-LLM, which achieves high accuracy (Macro F1 – 0.9115) in labelling reports using SFT and RLHF adapted to medical narratives. Both approaches have confirmed the effectiveness of hybrid supervised learning, but the current study is more versatile due to the integration of implicit feedback, while CheX-LLM specialises in medical tasks requiring the processing of complex descriptions.

The study demonstrated that continuous online learning for LLMs reduces the loss function from 3.82

to 3.15 in 500 steps, ensuring stable adaptation without signs of overlearning. For its part, the work of S. Pratap *et al*. (2025) proposed a decomposition of LLMs to analyse the fine-tuning contribution, which reveals a decrease in Tuning Contribution in successful attacks, highlighting the impact of overfitting on the model's vulnerability. The strength of the current approach is automated adaptation to diverse feedback in real time, but the weakness is the limited number of iterations and the risk of poor-quality feedback, which limits the potential and the balance between speed and stability. At the same time, the positive aspect of the study is the accurate analysis of the impact of fine-tuning on model responses, but the weakness is the narrow focus on attacks, which limits its versatility.

The study determined that online learning for LLMs ensures model stability (cosine similarity – 0.3964) thanks to EWC, but the low Relevance metric indicates the need for improved feedback filtering. In comparison, A. Hilel *et al*. (2025) demonstrated that feedback manipulation can introduce false knowledge into LLMs, increasing vulnerability to attacks. Both approaches emphasised the importance of feedback management, but the current study adapts to a variety of real-time queries, while the above focuses on specific attack scenarios, which limits their applicability. The OpenAssistant-based GPT-2 retraining improved the quality of responses over 500 fine-tuning steps, as evidenced by the increase in ROUGE-L, USS, and stability without signs of degradation. Instead, E. Mazzullo & O. Bulut (2025) studied the retraining of GPT-3.5-turbo for automated generation of feedback to open-ended questions, which provided high user satisfaction (84.8%) and structural quality of answers (72.9%), although some of them contained linguistic inaccuracies and generalisations. Both approaches confirmed the effectiveness of GPT model retraining in improving the relevance of generation, but the current study demonstrates a higher adaptability to the flow of real online feedback, which ensures not only learning from generalised patterns but also adapting to specific changes in user behaviour.

The results obtained showed that the developed Python software implementation is suitable for productive scenarios where rapid adaptation to real user feedback is substantial with limited resources. B. Yang *et al*. (2025) proposed the Multi-Objective Repair (MORepair) method, which aims to improve the performance of LLMs in software code repair tasks through training that covers both syntactic transformations and change logic. Their results demonstrated an increase in accuracy of up to 56% but require structured data and significant computational resources. In comparison, the current system is characterised by lower computational complexity, higher flexibility to unstructured feedback, and real-time readiness, making it suitable for dynamic LLM interaction environments. This study has shown that the developed GPT-2 online learning system is effective for adapting to real feedback in conditions of limited resources and an incomplete learning cycle. Instead, the study by S.V. Gaikwad *et al*. (2024) addressed retraining of the Bidirectional Encoder Representations from Transformers and GPT-3 models to improve the accuracy and stability of emotional tone analysis. Both approaches have shown the advantages of fine-tuning in specific tasks, but the architecture proposed in the current work is better adapted to unpredictable changes in user behaviour, in contrast to tasks where the feedback structure is more homogeneous, as in sentiment analysis.

While this study demonstrated that LLMs can adapt to diverse user feedback, increasing the relevance of responses while maintaining stability, S. Karunakaran & A. Jain (2025) investigated retraining LLMs to maintain a consistent personality of AI assistants using interaction history, context, and adaptive algorithms with a focus on emotional intelligence. Both approaches emphasised the importance of adapting to user needs, but the current study stands out for its integration of implicit feedback and fast real-time adaptation, making it more versatile for dynamic environments, while another study addressed building a personalised personality that is better suited for specific scenarios, such as chatbots. The results also demonstrated that the use of RLHF in combination with SFT for continuous online learning for LLMs can be used to adapt the model to a variety of feedback, improving the relevance of answers. In turn, the study by A. Bodaghi *et al*. (2024) employed RLHF with Proximal Policy Optimiser to generate 122.951 toxic samples, which significantly improved the performance of toxicity classifiers. Both approaches have confirmed the effectiveness of RLHF for adapting LLMs, but the current study addressed universal adaptation to real-world feedback in dynamic environments, while the aforementioned ones emphasise the specific task of generating balanced data for toxicity detection.

While this study has shown that the cosine similarity and dynamics of the loss function effectively illustrates the stability and progress of the GPT-2 model adaptation. X.-K. Wu *et al*. (2025) proposed the concept of Tutorial Fine-Tuning. It integrates hermeneutical theories to further describe adaptation of LLMs, emphasising the cognitive aspects of fine-tuning to achieve a higher level of intelligence. Both approaches emphasised the importance of adapting LLMs, but the current study features a practical real-time implementation with low computational complexity, while the above focuses on theoretical determination of fine-tuning, which has less practical applicability in dynamic scenarios.

Overall, this work showed that online training of LLMs ensures adaptation to diverse user feedback with improved response quality and model stability. In comparison, K. Wang *et al*. (2025) proposed the Learning through Communication (LTC) approach, which uses a universal buffer to store linguistic and non-linguistic feedback, improving the performance of LLM agents by 3.6-12% in various tasks. Both approaches have

demonstrated the effectiveness of LLMs in adapting to dynamic conditions, but the current study is distinguished by its low computational complexity and integration of implicit feedback, while LTC focuses on optimising agents in specific environments with higher performance metrics.

The study demonstrated that the use of a hybrid RLHF approach in combination with SFT can be used to adapt GPT-2 to a variety of real-time user feedback, increasing the relevance of answers by processing both explicit and implicit feedback. Regarding the study by C. Ye *et al*. (2025), an RLHF framework with a general preference oracle that avoids the assumptions of the Bradley-Terry model and applies inverse Kullback-Leibler divergence regularisation to optimise the policy was developed, which provided an advantage over other LLMs in general preference problems. Both approaches have highlighted the flexibility of RLHF for model adaptation, but the current study is prominent for its practical implementation of continuous online learning with a focus on handling unstructured feedback in dynamic environments, while the previous study focused on theoretical versatility and algorithmic efficiency for offline and online scenarios. Compared to the study by U. Kamath *et al*. (2024), which highlighted the challenges of RLHF, the need for many human evaluations and limited scalability due to the use of multiple LLMs as reward models, the current study favoured the integration of implicit feedback for continuous online learning of LLMs. This reduces the loss function and provides a moderate improvement in quality metrics with low computational complexity. However, it is also inferior in accuracy to specialised approaches such as Constitutional AI or Direct Preference Optimisation, which solve the problem of RLHF scalability.

While the study demonstrated the ability of a continuous online learning system to adapt to diverse user feedback, increasing the relevance of answers and maintaining model stability with a limited number of iterations, the study by E. Watson *et al*. (2024) proposed a new architecture for personalising LLMs through a token-based system for addressing individual values and context. The current approach is notable for its ease of implementation and low computational complexity, which makes it more suitable for dynamic productive environments and better able to handle unstructured feedback in real time. However, the system provided deeper personalisation due to its inductive approach and user-friendly interfaces for custom annotation.

The findings of this paper highlighted that online learning for LLMs provides adaptation to diverse user feedback, improving the quality of answers and reducing the loss function, while maintaining model stability. Similarly, the study by Q. Ai *et al*. (2024) addressed the improvement of Generative Information Retrieval through continuous learning methods and processing different types of feedback, such as contextual learning in dialogues and prompt learning. However, the current approach is notable for its simplicity of implementation and ability to efficiently process unstructured feedback in real time, while the methods may be more accurate in specialised information retrieval tasks. In summary, the current study provided fast online LLMs with low computational complexity, effectively adapting to unstructured real-time feedback. Its strengths are versatility and stability, but its weakness is the limited number of iterations, which limits the quality. In contrast to specialised methods, the current approach is better suited to dynamic productive environments.

CONCLUSIONS

The developed architecture of continuous online learning for LLMs integrates a multi-level adaptation cycle based on the processing and aggregation of both explicit and implicit user feedback. The implementation of classification, weight aggregation and signal buffering can be used to form of relevant query-expected response training pairs for further hybrid retraining (SFT and RLHF) with quality control using stability, relevance and A/B testing metrics. The results confirmed that the implicit feedback significantly exceeds the explicit feedback, which determines the priority of improving the classification and weight aggregation of these signals. The proposed architecture was implemented in scalable systems with support for parallel processing, which ensures real-time updating of LLMs without loss of stability or performance, as demonstrated by the NinjaTech AI example using customised fine-tuned models.

The software implementation of the LLMs online training cycle has demonstrated effectiveness in building a scalable pipeline based on real user feedback. Two key stages have been implemented in Python: the formation of a buffer of training pairs with weighting coefficients based on the OpenAssistant open data set and further training of the GPT-2 model using the Transformers library. In the process of data preparation, 21,265 query-expected-response examples were generated, incorporating relevance and length filters, and weights were determined based on user interaction. Fine-tuning GPT-2 on 52,912 examples with a limit of 500 steps ensured a steady decrease in the loss function from 3.82 to 3.15, without any signs of degradation or instability. The results showed that the model can effectively adapt even with partial retraining, demonstrating the advantages of the online approach over static and offline models.

A comprehensive evaluation of the results of adaptive re-training of LLMs showed a moderate but stable improvement in the quality of GPT-2 model answers after 500 fine-tuning steps on real user feedback. The automatic metrics (ROUGE-L = 0.1019, BLEU = 0.0060) showed an increase in lexical and semantic correspondence with the expected answers, and the UX metrics (USS = 0.1883) indicated the potential for further optimisation through better feedback sampling. The dynamics of learning loss confirmed the effectiveness

of the pipeline, and the cosine similarity analysis (mean value 0.3964) showed that the basic knowledge was retained with partial changes in answers, which indicates a balance between adaptation and stability. Examples of transformations of answers confirmed the growth of content, specificity and personalisation.

However, the study has certain limitations, such as the limited amount of feedback, the use of only one model architecture (GPT-2), and the reduced number of retraining steps. This reduces the possibility of generalising the results to more modern LLMs and larger application scenarios. Further research should focus on testing the approach at LLMs, expanding the feedback base, and studying the impact of different types of feedback on the quality of adaptation in the long term.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Ai, Q., Dou, Z., & Zhang, M. (2024). Improving generative information retrieval systems based on user feedback. In R.W. White & C. Shah (Eds.), *Information access in the era of generative AI* (pp. 111-133). Cham: Springer. doi: 10.1007/978-3-031-73147-1_5.

[2] Anisuzzaman, D.M., Malins, J.G., Friedman, P.A., & Attia, Z.I. (2025). Fine-tuning LLMs for specialized use cases. *Mayo Clinic Proceedings Digital Health*, 3(1), article number 100184. doi: 10.1016/j.mcpdig.2024.11.005.

[3] Balaskas, G., Papadopoulos, H., Pappa, D., Loisel, Q., & Chastin, S. (2025). A framework for domain-specific dataset creation and adaptation of large language models. *Computers*, 14(5), article number 172. doi: 10.3390/computers14050172.

[4] Bodaghi, A., Fung, B.C.M., & Schmitt, K.A. (2024). AugmenToxic: Leveraging reinforcement learning to optimize LLM instruction fine-tuning for data augmentation to enhance toxicity detection. *ACM Transactions on the Web*. doi: 10.1145/3700791.

[5] Cardó, A.V. (2025). *Investigating feedback types in reinforcement learning with human feedback and large language models*. (Bachelor's thesis, Aalto University, Espoo, Finland).

[6] Dombi, J., & Jónás, T. (2022). Weighted aggregation systems and an expectation level-based weighting and scoring procedure. *European Journal of Operational Research*, 299(2), 580-588. doi: 10.62441/nano-ntp.vi.5447.

[7] Gaikwad, S.V., Agarkar, P., Mohapatra, S., & Bagade, S. (2024). Fine-tuning LLM for sentiment analysis. *Nanotechnology Perceptions*, 20(6), 4946-4959. doi: 10.62441/nano-ntp.vi.5447.

[8] Haider, Z., Rahman, H., Devabhaktuni, V., Moeykens, S., & Chakraborty, P. (2025). A framework for mitigating malicious RLHF feedback in LLM training using consensus-based reward. *Scientific Reports*, 15, article number 9177. doi: 10.1038/s41598-025-92889-7.

[9] Hao, S., & Duan, L. (2025). Online learning from strategic human feedback in LLM fine-tuning. In *CASSP 2025 – 2025 IEEE international conference on acoustics, speech and signal processing* (pp. 1-5). Hyderabad: IEEE. doi: 10.1109/ICASSP49660.2025.10887891.

[10] Hilel, A., Shenfeld, I., Andreas, J., & Choshen, L. (2025). LLM hypnosis: Exploiting user feedback for unauthorized knowledge injection to all users. *ArXiv*. doi: 10.48550/arXiv.2507.02850.

[11] Kamath, U., Keenan, K., Somers, G., & Sorenson, S. (2024). Tuning for LLM alignment. In *Large language models: A deep dive* (pp. 177-218). Cham: Springer. doi: 10.1007/978-3-031-65647-7_5.

[12] Kampelopoulos, D., Tsanousa, A., Vrochidis, S., & Kompatsiaris, I. (2025). A review of LLMs and their applications in the architecture, engineering and construction industry. *Artificial Intelligence Review*, 58, article number 250. doi: 10.1007/s10462-025-11241-7.

[13] Karunakaran, S., & Jain, A. (2025). Fine-tuning LLMs for personality preservation in AI assistants. *International Journal of Research in Modern Engineering & Emerging Technology*, 13(4), 172-191. doi: 10.63345/ijrmeet.org.v13.i4.10.

[14] Lin, X., Wang, W., Li, Y., Yang, S., Feng, F., Wei, Y., & Chua, T. (2024). Data-efficient fine-tuning for LLM-based recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 365-374). New York: Association for Computing Machinery. doi: 10.1145/3626772.3657807.

[15] Martin, D. (2025). Improving CXR report labeling through LLM fine-tuning and human feedback. *Preprints*. doi: 10.20944/preprints202504.1668.v1.

[16] Mazzullo, E., & Bulut, O. (2025). Automated feedback generation for open-ended questions: Insights from fine-tuned LLMs. In *Proceedings of large foundation models for educational assessment* (pp. 103-120). Vancouver: MLResearchPress.

[17] Next Electronics. (n.d.). *Elastic weight consolidation (EWC)*. Retrieved from https://www.next.gr/ai/deep-learning-theory/elastic-weight-consolidation-ewc/.

[18] NinjaTech AI. (n.d.). *Ninja LLM suite*. Retrieved from https://www.ninjatech.ai/product/ninja-llm.
[19] OpenAssistant conversations dataset. (n.d.). Retrieved from https://huggingface.co/datasets/OpenAssistant/oasst1.
[20] Pratap, S., Aranha, A.R., Kumar, D., Malhotra, G., Iyer, A.P.N., & Shylaja, S.S. (2025). The fine art of fine-tuning: A structured review of advanced LLM fine-tuning techniques. *Natural Language Processing Journal*, 11, article number 100144. doi: 10.1016/j.nlp.2025.100144.
[21] Punnaivanam, M., & Velvizhy, P. (2024). Contextual fine-tuning of language models with classifier-driven content moderation for text generation. *Entropy*, 26(12), article number 1114. doi: 10.3390/e26121114.
[22] Rawal, N., Tavva, P., & Selvakumar, P. (2024). Enhancing large language model performance with reinforcement learning from human feedback: A comprehensive study on Q&A, summarization, and classification. In *2024 international conference on electrical, computer and energy technologies* (pp. 1-6). Sydney: IEEE. doi: 10.1109/ICECET61485.2024.10698396.
[23] Rehan, S., Al-Bander, B., & Al-Said Ahmad, A. (2025). Harnessing large language models for automated software testing: A leap towards scalable test case generation. *Electronics*, 14(7), article number 1463. doi: 10.3390/electronics14071463.
[24] Shi, T., *et al.* (2024). WildFeedback: Aligning LLMs with *in-situ* user interactions and feedback. *ArXiv*. doi: 10.48550/arXiv.2408.15549.
[25] Terras, N., Pereira, F., Silva, A.R., Santos, AA., Lopes, A.M., Silva, A.F.D., Cartal, L.A., Apostolescu, T.C., Badea, F., & Machado, J. (2025). Integration of deep learning vision systems in collaborative robotics for real-time applications. *Applied Sciences*, 15(3), article number 1336. doi: 10.3390/app15031336.
[26] Wang, J., Lu, H., Liu, Y., Ma, H., Wang, Y., Gu, Y., Zhang, S., Han, N., Bi, S., Baugher, L., Chi, E.H., & Chen, M. (2024). LLMs for user interest exploration in large-scale recommendation systems. In T. di Noia, P. Lops, T. Joachims. K. Verbert, P. Castells, Z. Dong & B. London (Eds.), *Proceedings of the 18th ACM conference on recommender systems* (pp. 872-877). New York: Assoviation Computing Machinery. doi: 10.1145/3640457.3688161.
[27] Wang, K., Lu, Y., Santacroce, M., Gong, Y., Zhang, C., & Shen, Y. (2025). Adapting LLM agents with universal communication feedback. In *Findings of the association for computational linguistics* (pp. 6105-6122). Albuquerque: Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.339.
[28] Watson, E., Viana, T., Zhang, S., Sturgeon, B., & Petersson, L. (2024). Towards an end-to-end personal fine-tuning framework for AI value alignment. *Electronics*, 13(20), article number 4044. doi: 10.3390/electronics13204044.
[29] Wu, X.-K., *et al.* (2025). LLM fine-tuning: Concepts, opportunities, and challenges. *Big Data and Cognitive Computing*, 9(4), article number 87. doi: 10.3390/bdcc9040087.
[30] Yang, B., Tian, H., Ren, J., Zhang, H., Klein, J., Bissyandé, T.F., Le Goues, C., & Jin, S. (2025). MORepair: Teaching LLMs to repair code via multi-objective fine-tuning. *ACM Transactions on Software Engineering and Methodology*. doi: 10.1145/3735129.
[31] Ye, C., Xiong, W., Zhang, Y., Dong, H., Jiang, N., & Zhang, T. (2025). Online iterative reinforcement learning from human feedback with general preference model. In *Proceedings of the 38th international conference on neural information processing systems* (pp. 81773-81807). Red Hook: Curran Associates Inc.
[32] Zhao, Q., Harper, F.M., Adomavicius, G., & Konstan, J.A. (2018). Explicit or implicit feedback? Engagement or satisfaction? A field experiment on machine-learning-based recommender systems. In *Proceedings of the 33rd annual ACM symposium on applied computing* (pp. 1331-1340). New York: Association for Computing Machinery. doi: 10.1145/3167132.3167275.

Безперервні цикли зворотного зв'язку: онлайн-налаштування LLM за допомогою сигналів користувача

Софія Швець

Магістр, спеціаліст з обробки даних
NinjaTech AI
94022, вул. Ель Каміно Реал, 4410, м. Лос-Алтос, Сполучені Штати Америки
https://orcid.org/0009-0000-7896-6479

Анотація. Інтенсивне зростання використання мовних моделей реального часу вимагає механізмів їх динамічної адаптації до змін у запитах, термінології та очікуваннях користувачів. Метою дослідження було вивчення підходів до перенавчання великих мовних моделей на основі безперервного зворотного зв'язку. Для досягнення цієї мети було застосовано теоретичне та структурно-функціональне моделювання архітектури адаптації, експериментальну реалізацію циклу перенавчання мовної моделі з обробкою та класифікацією різних типів зворотного зв'язку, а також кількісну оцінку результатів за допомогою автоматичних та користувацьких метрик. Результати дослідження показали ефективність архітектури безперервного онлайн-навчання, яка забезпечує актуальність та стабільність мовної моделі в реальному часі. У дослідженні визначено, що неявний зворотний зв'язок зустрічається в 4-10 разів частіше, ніж явний зворотний зв'язок, але явний зворотний зв'язок дає вищий приріст точності відповідей. Запропонована система успішно інтегрує різні типи користувацьких сигналів, забезпечуючи динамічну генерацію навчальних прикладів та гібридне перенавчання, зберігаючи при цьому якість та узгодженість результатів. Програмний цикл Python для адаптивного перенавчання мовної моделі включав обробку та фільтрацію користувацьких сигналів для формування високоякісного буфера навчальних пар. Після 500 кроків перенавчання на 52 912 парах запит-відповідь спостерігалося значне покращення моделі, що підтверджувалося зменшенням функції втрат з 3,82 до 3,15 та стабільністю процесу точного налаштування без ознак перенавчання. Результати попереднього навчання показали помірне покращення якості відповідей після адаптації: лексична подібність за даними Recall-Oriented Understudy for Gisting Evaluation становила 0,102, точність за даними Bilingual Evaluation Understudy – 0,006, а суб'єктивна задоволеність користувачів зросла до 0,24, зберігаючи при цьому стабільність моделі із середнім значенням косинусної подібності 0,396. Підхід, запропонований у цьому дослідженні, покращує якість та релевантність відповідей мовних моделей у реальному часі, зберігаючи їх стабільність, і може бути використаний у продуктивних системах для покращення користувацького досвіду

Ключові слова: адаптивне перенавчання; генеративні трансформатори; адаптація динамічної моделі; реалізація на Python; гібридне навчання; метрики оцінки якості; стабільність мовної моделі

BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 30 No. 3. 2025

UDC 004.056.5:004.42:004.942.5
DOI: 10.62660/bcstu/3.2025.121

Adaptive hybrid SMS spam detection system with user feedback-based self-learning

Ihor Liakh

Doctor of Technical Sciences, Professor
Uzhhorod National University
88000, 3 Narodna Sqr., Uzhhorod, Ukraine
https://orcid.org/0000-0001-5417-9403

Andriy Chorniy*

PhD in Technical Sciences, Associate Professor
Cherkasy State Technological University
18006, 460 Shevchenka Blvd., Cherkasy, Ukraine
https://orcid.org/0000-0002-0989-7112

Oksana Lutak

Master's Student
Uzhhorod National University
88000, 3 Narodna Sqr., Uzhhorod, Ukraine
https://orcid.org/0009-0002-0798-6928

Marian Tsenkner

Master's Student
Uzhhorod National University
88000, 3 Narodna Sqr., Uzhhorod, Ukraine
https://orcid.org/0009-0003-2752-7398

Abstract. This study presented a comprehensive approach to SMS spam detection based on a hybrid architecture that integrated local message processing algorithms with high-performance cloud-based deep learning models. This approach enabled a balance between classification accuracy and the privacy of processed messages. The objective of this study was to develop an intelligent hybrid SMS spam detection system capable of delivering high classification accuracy, maintaining up-to-date knowledge, enabling user personalisation, and adapting to new attack patterns. To achieve the study's objective, a comprehensive analytical approach was applied, combining a detailed review of scientific literature on SMS spam detection – including machine learning, neural networks, and hybrid methods – with empirical analysis. To implement classic machine learning models (Naïve Bayes, Logistic Regression, Random Forest), standard machine learning libraries were used, and for deep learning, frameworks that support recurrent neural networks, in particular Long Short-Term Memory and transformer architectures, were applied. The system was tested on the open SMS Spam Collection dataset using Accuracy (up to 0.98), F1-score (up to 0.95) and ROC-AUC (up to 0.98) metrics. Moreover, a system was developed to dynamically update knowledge based on user feedback, alongside a weighted framework designed to evaluate the trustworthiness of that feedback. During the study, a multi-level system was developed that performed initial classification on the user's device with the ability to delegate processing to a cloud module in cases

Article's History: Received: 28.03.2025; Revised: 01.08.2025; Accepted: 15.09.2025.

Suggested Citation:
Liakh, I., Chorniy, A., Lutak, O., & Tsenkner, M. (2025). Adaptive hybrid SMS spam detection system with user feedback-based self-learning. *Bulletin of Cherkasy State Technological University*, 30(3), 121-132. doi: 10.62660/bcstu/3.2025.121.

*Corresponding author

of uncertainty. Compared to basic approaches, the hybrid architecture demonstrated improved classification accuracy, reduced false positives and false negatives, and increased adaptability to changes in the structure of spam messages. Aggregation of suspicious messages in the cloud ensured effective retraining of models in cases of conceptual shift. The practical value of the results lies in the potential integration of the developed system into mobile platforms, as well as corporate information security tools, for the purpose of filtering SMS content and protecting end-users from social engineering

Keywords: natural language processing; long short term memory; architecture of spam; messages; metric

INTRODUCTION

The relevance of the study was determined by the rapid increase in the number of malicious Short Message Service (SMS) messages, which were used as tools of phishing, social engineering, and fraud. The complication of spam text structures, the emergence of disguised messages, and the use of bypassing techniques by attackers significantly reduced the effectiveness of traditional filtering methods. At the same time, the need for the development of intelligent systems grew, as they had to ensure high classification accuracy while preserving user data confidentiality. Increased requirements for the speed, adaptability, and scalability of message processing created the demand for new architectures that combined local processing with cloud computing and supported mechanisms of continuous knowledge updating in the dynamic environment of mobile threats.

Research devoted to the detection of mobile SMS spam, considerable attention was paid to the application of hybrid systems, text vectorisation methods, and the use of natural language processing (NLP). One of the promising directions was the use of federated learning, which allowed the preservation of user privacy without a loss in model accuracy. The scientists S. Vats *et al*. (2024) studied SMS spam classification within the framework of federated learning, emphasising the balance between model accuracy and the preservation of user privacy. In the proposed approach, message processing took place directly on the user's device, while the already trained models were combined without transferring personal data to a central server. The study results demonstrated the high efficiency of the method: classification accuracy reached 98.3% as early as the third training stage, which confirmed the stability and accuracy of the model while maintaining confidentiality. In addition, a gradual decrease in the loss function was observed from the first to the tenth epoch, which proved the effectiveness of the learning process. The authors gave insufficient attention to the issue of model scalability under limited computational resources and did not address the problems of concept drift and adaptation to dynamically changing spam patterns, which were critically important for the practical implementation of hybrid systems.

Hybrid SMS spam detection systems, which combined several machine learning (ML) algorithms, continued to demonstrate high adaptability to new attack scenarios. The researchers H. Baaqeel & R. Zagrouba (2020) investigated the problem of classifying SMS messages as spam or "ham" using ML methods. In their study, they proposed a hybrid system that integrated supervised and unsupervised learning algorithms to improve the accuracy of unwanted message detection. The authors concluded that the application of a combined approach enhanced the system's performance, particularly in terms of accuracy and the F-measure. Simultaneously, the study devoted insufficient attention to the aspects of system adaptability to new types of attacks and the possibility of dynamic knowledge updating, which were essential for counteracting the rapid evolution of spam techniques.

Researchers A. Al Maruf *et al*. (2023) developed an ensemble-based approach to the classification of SMS messages in Bengali (Bangla), which enables the detection of spam even with a relatively small training dataset. The study employed logistic regression, decision trees, support vector machines, and random forests as baseline models, with the results subsequently enhanced through ensemble techniques such as bagging, boosting, and stacking. The authors concluded that the ensemble methodology, particularly the XGBoost model, achieved the highest classification accuracy for Bengali SMS. However, the study provided insufficient discussion regarding the scalability of the proposed approach to larger datasets and other languages, as well as the robustness of the system against emerging types of spam, which limits the potential for its broader application.

The researchers M.A. Abid *et al*. (2022) focused on the importance of high-quality text feature preparation in classifying SMS messages as spam or "ham" using classical ML algorithms. In the study, the bag-of-words and term frequency-inverse document frequency (TF-IDF) methods were applied for feature construction, and over-sampling and under-sampling strategies were employed to balance the uneven dataset. The experimental results showed that the Random Forest algorithm delivered the highest classification accuracy, reaching 99% for both SMS and a spam email corpus. The authors concluded that proper feature engineering played a key role in the performance of the models. Concurrently, the study gave insufficient attention to the generalisation of the approach to multilingual corpora and to its effectiveness in handling dynamically changing data, which limited the practical applicability of the developed system.

The researchers D.A. Oyeyemi & A.K. Ojo (2024) proposed an approach to SMS spam detection based on

NLP methods combined with ML models. In the study, data preprocessing (removal of stop words, tokenisation) and vectorisation using Bidirectional Encoder Representations from Transformers (BERT) were implemented. Classification was carried out with several algorithms, including support vector machine (SVM), logistic regression, the Naïve Bayes classifier, gradient boosting, and random forest. The results showed that the combination of Naïve Bayes and BERT achieved the highest accuracy (97.31%) with minimal execution time (0.3 seconds), which indicated the high efficiency of the approach and a low rate of false positives. The authors concluded that the integration of BERT with classical algorithms was capable of significantly improving the quality of SMS spam filtering. Contemporaneously, the study devoted insufficient attention to the scalability of the solution for large datasets and to its applicability in multilingual environments, which might limit the practical use of the proposed system.

The researchers S. Gadde *et al*. (2021) investigated the application of ML and deep learning (DL) methods for SMS spam detection. In the study, the UCI dataset was used, on which the authors compared the performance of classical ML algorithms with recurrent neural networks using long short-term memory (LSTM). The experimental results showed that the LSTM model achieved an accuracy of 98.5%, outperforming the other tested algorithms, which confirmed the relevance of integrating DL into spam filtering systems. The authors concluded that the combination of ML and DL methods could significantly improve system effectiveness in real-world conditions. In parallel, the study gave limited consideration to the optimisation of the LSTM architecture for different corpus sizes and did not analyse the computational resource costs, which were important for the practical implementation of such models.

The additional difficulties in combating spam arose from the methods employed by attackers to bypass detection systems. M. Salman *et al*. (2024) conducted a comparative analysis of various ML models and contemporary anti-spam services using a new large dataset of over 68,000 SMS messages, of which 39% were spam. The study included both shallow ML methods and deep neural networks, as well as an analysis of the models' robustness to evolutionary and covert attacks by spammers. The results showed that most classical ML solutions and existing anti-spam services demonstrated insufficient effectiveness against bypass strategies. In tandem, the authors concluded that the combination of classical and modern models could enhance system resilience to such attacks, opening avenues for the development of more reliable SMS spam filters.

In turn, V.V. Kalyani *et al*. (2024) demonstrated the effectiveness of recurrent neural networks (RNN) for SMS spam classification. Using a dataset of 5,570 messages, the authors first addressed the class imbalance problem using the ADASYN method and applied TF-IDF vectorisation for text representation. Baseline ML algorithms, including Random Forest, achieved an accuracy of 92.3%, whereas RNN architectures significantly outperformed these results: LSTM reached 98%, Bi-LSTM 98.2%, and GRU 98.3%. The highest performance was delivered by a hybrid model combining Bi-LSTM and GRU, achieving 99% accuracy. The authors' contribution lay in demonstrating the advantages of DL models over classical ML approaches, as well as in the use of data balancing methods, which enabled maximum accuracy even with a limited dataset size. Therefore, the analysis of current literature confirmed the relevance of developing hybrid, adaptive, and privacy-oriented approaches to SMS spam detection, combining the advantages of ML, NLP, and continuous knowledge updates. The aim of this study was to develop an adaptive SMS spam detection system that integrates both local and cloud-based classification models, enabling continuous knowledge updates.

MATERIALS AND METHODS

The study was conducted within the framework of developing an intelligent system capable of effectively countering SMS spam with dynamic patterns, including phishing messages (smishing). The methodological foundation of the study was based on the principles of NLP, adaptive text data analysis, and dynamic knowledge updating. For empirical testing of the proposed approach's effectiveness, the publicly available SMS Spam Collection Dataset was used, containing 5,574 SMS messages that had been pre-classified as either spam or legitimate. The effectiveness of the proposed SMS spam detection system was evaluated through a series of experiments using publicly available datasets. The primary testing platform was the SMS Spam Collection Dataset, which contained 5,574 labelled examples of real text messages, including 4,827 legitimate (ham) messages and 747 spam messages (Almeida & Hidalgo, 2011).

During the text data preprocessing stage, standard operations were performed, including normalisation, removal of noise elements, tokenisation, and the transformation of messages into vector-based features using frequency-statistical methods (e.g., TF-IDF) and semantic models. At the local level, a pre-trained model was applied to instantly classify incoming messages as spam or legitimate. In cases of uncertainty or the detection of new patterns, the system could initiate forwarding the message to the cloud for re-analysis using a more resource-intensive but more accurate model. Contemporary DL methods were employed for this purpose, particularly LSTM models, which had demonstrated high effectiveness in text classification tasks (Gomaa, 2020). The training of such models was often based on the minimisation of a loss function, for example, the cross-entropy function:

$$CrossEntropy(p, q) = -\sum_n p_n \times ln(q_n), \quad (1)$$

where p – the true label and q – the predicted value from the model's output.

For the construction of classification models, both classical statistical algorithms (Naïve Bayes, Logistic Regression) and more complex ensemble and DL models (Random Forest, LSTM) were employed, allowing a comparative analysis of their effectiveness. Each model was evaluated using the primary metrics for binary classification: accuracy, precision, recall, F1-score, and ROC-AUC. These metrics enabled a comprehensive assessment of the models' ability to effectively detect SMS spam while minimising false-positive classifications. Subsequently, the vector representation was sent to a local filter implemented using a ML model (e.g., Naïve Bayes or Logistic Regression) according to the study of D. Bäckman (2019). For the Naïve Bayes method, the probability that a message belonged to a certain class C given the presence of features X is determined by the following formula:

$$P(C|X) = \frac{P(X|C)*P(C)}{P(X)}, \quad (2)$$

where $P(C|X)$ – the *posterior* probability of the class, $P(X|C)$ – the likelihood, $P(C)$ – the prior probability of the class, and P(X) is the probability of the features.

For Logistic Regression, the logistic function (sigmoid) was used, which transforms a linear combination of features z into a value within the range (0.1):

$$\Lambda(z) = \frac{1}{1+e^{-z}}, \quad (3)$$

where $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$, β_i – model coefficients, x_i – feature values.

The architecture of the proposed system provided for local preliminary classification of messages, with the option to forward uncertain cases to the cloud model for enhanced analysis. The cloud model employed DL methods and performed periodic retraining based on new data and user feedback. The study implemented a mechanism for dynamic knowledge updating, considering user feedback, its reliability, and changes in the statistical characteristics of the data over time. To ensure stability, change buffers, update parameter controls, and anomaly aggregation were used. Table 1 provides the structure of a representative sample used during model retraining. The sample includes different message types, their quantities, the reliability of user feedback associated with each type, and the assigned sample weights used to prioritise model updates.

Table 1. Example structure of a representative sample for model retraining

Message Type	Number of Examples	Feedback Reliability (%)	Sample Weight
Correctly Classified (Spam)	350	95	1.0
Correctly Classified (Legitimate)	420	94	1.0
False Positives	60	88	1.3
False Negatives	45	90	1.3
Low Confidence Messages	80	85	1.2

Source: compiled by the authors

The dataset was based on accumulated user feedback, taking into account not only the type of classification (correct or incorrect) but also the reliability of the provided feedback. As shown in Table 1, the structure of the dataset included five main categories: correctly classified spam and legitimate messages, two types of classification errors (false positives and false negatives), and messages for which the model exhibited a low confidence level in its decision. For each category, the average reliability of the feedback was also indicated, reflecting the proportion of confirmed or highly probable evaluations, as well as the weight within the dataset, which determined the influence of the respective subset on the model update process. At the final stage, the effectiveness of the SMS spam classification models was evaluated using five key metrics: Accuracy, Precision, Recall, F1-score, and ROC-AUC. The obtained values were summarised in an overall performance vector:

$$E = \{e_1, e_2, e_3, e_4, e_5\}, \quad (4)$$

where each component corresponded to a separate metric. For the interpretation of results, a linguistic scale for evaluating model performance was introduced:

$e \in (0.8; 1]$ – l_1 = "high performance", $e \in (0.6; 0.8]$ – l_2 = "above-average performance", $e \in (0.4; 0.6]$ – l_3 = "average performance", $e \in (0.2; 0.4]$ – l_4 = "low performance", $e \in [0; 0.2]$ – l_5 = "very low performance".

To achieve the aim of the study, a comprehensive approach was employed, combining theoretical and empirical methods. The research was conducted using an analytical method, reviewing the scientific literature devoted to SMS spam detection, including the use of ML algorithms, neural networks. At the initial stage, a thorough review of scientific literature was conducted, focusing on SMS spam detection methods, particularly those involving ML algorithms, neural networks, and hybrid architectures that integrate various text processing techniques. The proposed hybrid architecture of the SMS spam detection system was built on the principle of distributed processing, combining a local level (on the user's device) and a cloud level (centralised processing). At the local level, basic message classification was performed with minimal latency. Messages with uncertain or suspicious results were automatically forwarded to

the cloud level, where more complex and resource-intensive DL models (such as LSTM) were applied. A key element of the architecture was the classification result weighting mechanism, which aggregated the outcomes of the local and cloud models, taking into account: the confidence level of each model, the historical accuracy of the models on similar messages, the trustworthiness of the message source (where available).

Model updates were carried out asynchronously, based on aggregated suspicious messages received from a large number of users. To prevent training on anomalous or deliberately manipulated data (data poisoning), a knowledge update stabilisation mechanism was implemented in the cloud component of the system; specifically: a message buffer with controlled size and time window was used, the consistency of the statistical characteristics of new data was checked (for example average message length, frequency of spam tokens), mechanisms for anomaly filtering and weight-limited updates were applied to prevent overfitting to exceptional cases.

The model retraining mechanism was based on periodic analysis of the accumulated message buffer containing user feedback. If the number of unique new patterns exceeded threshold values, controlled retraining of the model was initiated, with the previous parameters retained as the baseline configuration. The model selection was guided by the need to achieve an optimal trade-off between classification speed, resource efficiency, and spam detection accuracy, considering the functional distribution of the system across both local and cloud environments. Such a hybrid architecture addresses the limitations of computing power on user devices while at the same time enabling the implementation of more complex algorithms within the cloud environment.

At the local level, Naïve Bayes and logistic regression models were employed, as they were characterised by high processing speed, low resource consumption, and ease of implementation. This ensures rapid decision-making without delays and independently of network connectivity. Logistic regression, although still a linear model, provides greater sensitivity to individual features (such as distinctive words or phrases), which in turn helped to improve the overall balance of classification. However, both local models are limited in their ability to detect complex or hidden patterns, as they do not account for sequential dependencies between words and do not support the modelling of non-linear relationships. To overcome these limitations, more powerful algorithms such as Random Forest and LSTM are employed at the cloud level. Random Forest, as an ensemble of decision trees, made it possible to model non-linear relationships between features and demonstrates resilience against overfitting, even when working with complex or noisy data. Its effectiveness lies in its ability to generalise information across multiple trees and filter out random anomalies. The deployment of this model in the cloud environment has enabled the processing of large volumes of data with flexible parameter tuning.

The highest performance indicators were achieved with the LSTM model, which belonged to the class of recurrent neural networks and is specialised in processing sequences. The LSTM architecture permitted to retain the context of preceding words, which was critically important for spam classification, as such messages often contain manipulative sequences, dynamic patterns, or veiled calls to action. Owing to the computational capabilities of the cloud environment, this model can be trained on the full dataset without simplifying its structure, thereby ensuring high sensitivity and strong generalisation ability. For the local level, Naïve Bayes and logistic regression models were chosen due to their high processing speed and moderate accuracy. These algorithms had low computational requirements, allowing for rapid classification without significant delays on user devices, even when processing power is limited. In addition, they performed well with vector representations of short texts, such as TF-IDF, which makes them effective for the initial filtering of messages.

At the cloud level, more sophisticated models were employed – Random Forest and LSTM. Random Forest provides flexibility and reliability, particularly when the feature set is limited, as its ensemble nature allows it to model complex non-linear relationships and respond robustly to noisy data. LSTM was selected as the primary model for deep analysis because it can capture the sequential order of words in a message, a capability crucial for uncovering hidden meanings in phishing texts. Owing to its architecture, LSTM can retain context and model dependencies in long sequences, significantly improving the classification accuracy of complex and dynamic spam patterns. In this way, the combination of fast and lightweight models at the local level with powerful deep learning algorithms in the cloud provided an optimal balance between processing speed, accuracy, and the system's adaptability to changes in the nature of SMS spam.

RESULTS AND DISCUSSION

The proposed SMS spam detection system featured a hybrid architecture, combining local and cloud levels of message processing. This approach enabled a balance between classification accuracy and data privacy (Al-Zebari *et al.*, 2025). Because of this loss function, the model optimised its weights during training, which reduced the number of misclassifications of messages as spam or legitimate content. Its high sensitivity to the probability distribution between classes provided flexibility when handling ambiguous or evolving message patterns. Within the proposed system, this contributed to more accurate detection of new or modified SMS spam patterns, even in a dynamic information environment where attackers continually adapt message content to bypass filters.

At the cloud level, the system aggregated suspicious messages received from a large number of devices. These data were used for dynamic knowledge updating through model retraining. The collection of such messages could be carried out with anonymisation and in accordance with privacy policies described in federated learning studies (Li *et al.*, 2024). The cloud model performed periodic retraining based on the updated dataset, which maintained the relevance of vector representations and prevented shifts in the statistical characteristics of the data over time, as is typical for SMS spam.

Dynamic updating was ensured through continuous learning mechanisms, allowing the model to adapt to new threats in real time (Boyko & Kovalchuk, 2023). Stability control techniques were applied, such as restricted parameter updates or the use of change buffers, which were introduced only after validation on test data. This approach prevented overfitting to anomalies and preserved the model's generalisation capability amid rapid changes in spam strategies. Retraining on anomalies can cause a model to memorise atypical or random patterns in the input data, reducing its ability to generalise and perform effectively on new examples. For instance, if a short burst of unusual but random messages is received and classified as spam, a model that updates without additional checks may adapt to these patterns and lose effectiveness in detecting more general forms of spam. Therefore, the use of change buffers and the validation of new parameters on a separate dataset helps determine whether the new data truly represents a sustained shift in patterns rather than a short-lived anomaly.

Preserving the model's generalisation capability was critically important in the context of the continual evolution of spam strategies. Spammers could periodically change keywords, message structure, or delivery methods, causing rapid shifts in the distribution of incoming data. The adaptive system needed to distinguish genuine emerging trends from random noise in the data. The application of update stabilisation methods prevented responses to isolated spikes, instead focusing adaptation on long-term patterns, thereby ensuring the robustness and relevance of the classification model in a dynamic environment. User or system feedback was also taken into account during the knowledge update process. This enabled the construction of an adaptive system capable of evolving in response to changes in the nature of SMS spam. Incorporating feedback supported not only the technical adaptation of models but also personalisation – the system could account for the individual perception of spam by different user groups. Similar architectures had already been examined in contemporary studies, for instance by S. Hossain *et al.* (2022) on multi-level spam filtering, confirming the effectiveness of multi-component systems with cyclical knowledge updating. Structurally, the operation of the proposed system was illustrated in Figure 1.

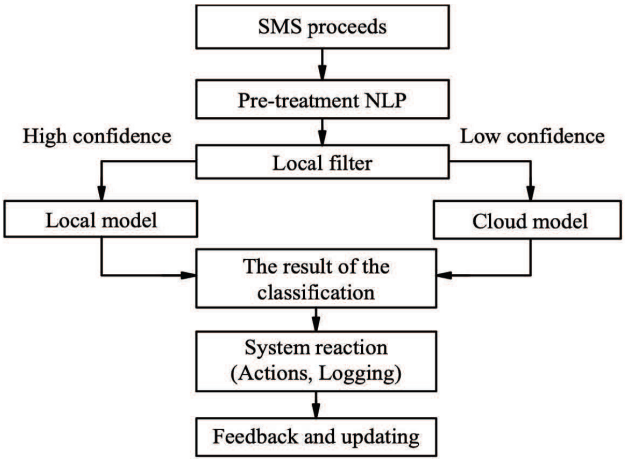


Figure 1. System operation flowchart: from message receipt to response
Source: compiled by the authors

The architecture of the proposed adaptive SMS spam detection system relied on the integration of local data processing with cloud technologies and supports the dynamic updating of knowledge models. The core process of handling incoming messages is implemented through a sequence of functional modules, each responsible for a specific task. The initial stage involved receiving an SMS message, which was then passed to a pre-processing module that utilises NLP techniques. That module was responsible for cleaning the text, normalising it, tokenising, and generating a vector representation of the message using approaches such as TF-IDF or word embeddings, as stated in the study by C.N. Mohammed & A.M. Ahmed (2024). In cases where the classifier demonstrated high confidence, the message was classified immediately at the local level. However, if the confidence level was insufficient, the message was redirected to the cloud-based model, which employed more advanced computational algorithms, such as BERT or LSTM, for more accurate classification as was proved in the research made by J. Prashob & S.Y. Yerima (2022). The classification results from both the local and cloud-based models were combined within the aggregated results module, which made the final decision regarding whether a message belongs to the spam category or to legitimate communications.

The next stage is the system's response, which involved carrying out the appropriate actions – filtering, notifying the user, or logging events. The final module was responsible for collecting feedback and updating the models through retraining, taking into account classification corrections based on feedback from users or administrators. The effectiveness of spam detection systems largely depended on their ability to adapt to new types of messages, which evolve over time. Within the proposed hybrid architecture, an adaptive knowledge update mechanism was introduced, ensuring the dynamic improvement of classification models based on user feedback. After classification, each message could

be evaluated by the user or system log as correctly or incorrectly classified. These responses were aggregated and stored in a dedicated feedback collection module. Subsequently, taking into account the quantity, type, and reliability of such feedback, a representative dataset was formed for retraining or further training of the models.

Assigning higher weights to incorrectly classified messages (1.3) and messages with low confidence (1.2) allowed the model to focus on the most critical cases that required revision of classification behaviour. Nevertheless, messages with high classification reliability and confirmed feedback were assigned a standard weight (1.0), ensuring stability in the training process and preventing overfitting on erroneous or noisy examples. This approach allowed a balance to be achieved between the model's adaptability to new types of messages and the stability of system behaviour in operational conditions, supporting the maintenance of high classification accuracy without the need for complete retraining of the model. To prevent shifts in the statistical characteristics of the data over time and avoid overfitting, quality filters were applied: messages with high classification confidence and confirmed feedback were assigned higher weights during the knowledge update process. Furthermore, updates were not instantaneous – a pre-aggregation approach was employed, which allowed the stability of the model to be controlled at the production level (Rojas-Galeano, 2021). This approach ensured the gradual adaptation of the system to changes in SMS spam patterns, maintaining high relevance and classification accuracy without the need for manual model review or complete retraining. Figure 2 presented a schematic representation of the system's dynamic update mechanism based on collected feedback:

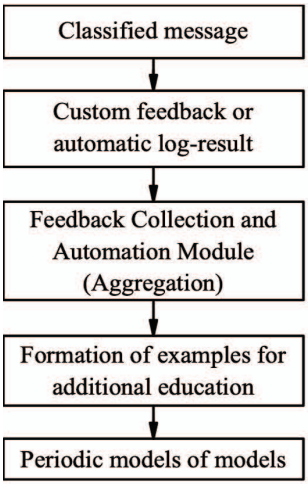


Figure 2. Adaptive knowledge update mechanism based on feedback
Source: compiled by the authors

The diagram illustrates the dynamic knowledge update workflow within the SMS spam detection system, based on user feedback. After each classification decision, users were given the opportunity to confirm or dispute the result. This feedback was collected and processed by evaluating source reliability, message recurrence, and other contextual features. An adaptive model was then constructed, which periodically updates both local and cloud-based components of the system. This feedback loop ensures that the model remains responsive to new spam behaviours, thereby mitigating the risk of performance degradation over time. To ensure an objective and comprehensive analysis of the effectiveness of the adaptive SMS spam detection system, a mechanism was developed that encompasses the complete cycle of constructing and evaluating classification models. At the initial stage, incoming SMS messages undergo pre-processing. This process includes text normalisation, which involves converting characters to lowercase and removing special characters, such as HTML tags or URLs. This was followed by tokenisation, where text had been split into individual words or tokens, and the removal of stop words that carry little semantic meaning. To further refine the textual data, stemming or lemmatisation techniques were applied to reduce words to their root forms. After these steps, the text was transformed into a numerical vector representation using approaches such as TF-IDF or modern embedding models.

To evaluate the performance of the adaptive SMS spam detection system, a set of quantitative metrics commonly applied in binary classification problems was employed. These metrics enable the assessment of not only the overall accuracy of the models but also their capability to identify spam messages while minimising false positive rates. Accuracy reflected the overall proportion of properly classified messages (both spam and legitimate) relative to the entire dataset. A high value of this metric indicates general consistency of the model; however, it may be insufficiently informative in cases of class imbalance, which is typical for spam filtering tasks. Precision characterised the proportion of truly spam messages among all messages classified as spam by the model. A high precision level indicates a low likelihood of false positives, which was critical in the context of avoiding the loss of legitimate messages. Recall, on the other hand, measured the model's ability to identify all existing spam messages within the dataset, including those that were more difficult to detect. This is particularly important in situations where missing harmful content was unacceptable from a security or user experience perspective.

The F1-score represented the harmonic mean of precision and recall, providing a balanced assessment of the model, which was particularly useful when both metrics are considered equally important for the task. A high F1-score indicated the presence of a classifier that was both accurate and sensitive. The Receiver Operating Characteristic – Area Under the Curve (ROC-AUC) was used to assess the model's effectiveness in differentiating between spam and legitimate messages

across varying classification thresholds. An AUC value approaching 1.0 indicates a strong ability to discriminate between classes. This metric was especially valuable in scenarios where accurate class separation was essential, as it encapsulates the trade-off between sensitivity and specificity by balancing false positive and false negative rates (Molina-Coronado *et al.*, 2023).

Precision and recall reflected opposite types of errors, and their balance is particularly important in cases of class imbalance. For example, when the "spam" class is rare, recall is more sensitive to how well the model detects this class, even if precision decreases as a result. Conversely, high precision combined with low recall indicates that the model detects only the "obvious" spam messages while ignoring more complex examples. The F1-score, as the harmonic mean of precision and recall, provides a more balanced assessment in situations of class imbalance. This was particularly useful when it is important to avoid both missing spam and misclassifying legitimate messages. ROC-AUC was invariant to class ratios, but in cases of strong imbalance it can sometimes be more useful to employ PR-AUC (Precision-Recall AUC), which was more sensitive to variations in the rare class. For example, when recall was high but precision was very low, this indicated that the model had been overly aggressive in labeling messages as spam, which could have resulted in the loss of important messages.

In such cases, even a high ROC-AUC did not guarantee the model's practical usefulness. Therefore, when analysing classification models in imbalanced-class scenarios, it was advisable to consider a set of metrics collectively and in light of the task's specific risk priorities. The comparative results of four different models, presented in Table 2, indicate a significant improvement in performance when using cloud-based algorithms employing ensemble and neural methods (Random Forest, LSTM) compared with classical statistical approaches (Naïve Bayes, Logistic Regression). The highest values for accuracy, F1-score, and ROC-AUC were achieved by the LSTM model, demonstrating the advantage of DL in tasks involving sequential text data processing. However, in tasks with a strong class imbalance – such as spam filtering, where legitimate messages predominate – accuracy can give a misleading impression of a model's effectiveness. Formally, if the "spam" class constitutes only 5% of all messages, even a "dumb" model that always predicts "not spam" would achieve an accuracy of approximately 95%, despite failing to detect any spam. Therefore, it is important to consider other metrics that more accurately reflect classification quality under such conditions.

Table 2. Metrics for evaluating the effectiveness of SMS spam classification models

Model	Accuracy	Precision	Recall	F1-measure	ROC-AUC
Naïve Bayes (local)	0.92	0.88	0.85	0.86	0.91
Logistic Regression	0.94	0.90	0.88	0.89	0.93
Random Forest (cloud)	0.97	0.95	0.93	0.94	0.97
LSTM (cloud)	0.98	0.96	0.95	0.95	0.98

Source: compiled by the authors

The results presented in Table 2 demonstrate a gradual increase in classification efficiency depending on the complexity of the model, its ability to generalise, and its ability to examine the context of incoming messages. The Naïve Bayes model, which assumed of conditional independence of features, demonstrated the lowest, though still acceptable, performance: accuracy was 0.92, precision 0.88, and recall 0.85. This level of effectiveness was typical for this model in text classification tasks, where the consideration of inter-word relationships was limited. The model was fast and straightforward to implement; however, it lacks the flexibility to adapt to more complex structures in the text, which partially explains the lower F1-score (0.86) and ROC-AUC (0.91) compared with other approaches. Naïve Bayes model, despite the assumption of conditional independence of features, demonstrates solid performance in short-text classification tasks when used with TF-IDF, achieving accuracy rates exceeding 96%.

Logistic Regression, which has a better capacity to model linear relationships between features, achieved higher performance: accuracy reached 0.94, precision 0.90, and recall 0.88. Despite the linearity of the model, it more effectively accounts for the weight of individual features (such as specific words or phrases characteristic of spam), allowing for a balanced F1-score (0.89) and AUC (0.93). However, the inability to model complex non-linear relationships and sequences somewhat limited its overall performance. Random Forest, as a representative of ensemble tree-based methods, demonstrated substantially higher performance: accuracy was 0.97, precision 0.95, recall 0.93, F1-score 0.94, and AUC 0.97. This is attributable to the model's ability to generalise even in cases of complex or noisy data. The use of multiple independent trees reduces the risk of overfitting and better captures the variability of spam patterns. In this experiment, Random Forest was deployed in a cloud environment, which allowed for the processing of larger data volumes and the utilisation of optimised parameters.

The highest performance was demonstrated by the LSTM model – a neural network specialised in sequence processing. Its accuracy reached 0.98, precision – 0.96, recall – 0.95, F1-score – 0.95, while the ROC-AUC achieved 0.98. Owing to its architecture, which is capable of retaining context at the word-sequence level,

LSTM proved most effective in recognising hidden patterns within spam messages, particularly those involving manipulative vocabulary, sequential instructions, or "technical" phrases (e.g., "click here", "you have won"). Furthermore, the cloud environment provided sufficient computational resources to train the deep model on the full dataset without compromising its complexity. The overall trend in performance indicates that the transition from simple statistical methods to DL approaches provided a substantial improvement in results, particularly in tasks with high semantic complexity and intricate contextual dependencies, which are characteristic of modern spam. Thus, all the models considered demonstrated high effectiveness (l_1) in the task of SMS spam classification; however, the most successful was the LSTM neural network operating within a cloud environment.

M. Ahmadi *et al*. (2025) investigated the effectiveness of various feature extraction methods and classification algorithms for detecting SMS spam. They compared six classical and deep classifiers – Naïve Bayes, K-Nearest Neighbours, Support Vector Machines, Linear Discriminant Analysis, Decision Trees, and Deep Neural Networks – in combination with two text vectorisation methods: bag-of-words and TF-IDF. The study results showed that the TF-IDF method significantly outperformed bag-of-words across all classifiers, and the combination of TF-IDF with Naïve Bayes achieved the highest accuracy of 96.2%. Compared with other models, such as Support Vector Machines and Deep Neural Networks, Naïve Bayes with TF-IDF demonstrated superior accuracy and a better balance between detecting spam and legitimate messages.

These findings underscored the importance of selecting an appropriate feature extraction method to enhance the effectiveness of a spam detection system. Compared with the study, which also employs Naïve Bayes at the local level with TF-IDF for fast and accurate initial classification, the results of this research support the choice of lightweight and fast models for preliminary filtering. However, unlike their approach, this hybrid method complements the local classification with more sophisticated cloud-based models, thereby improving the system's accuracy and adaptability to new spam patterns. At the cloud level, suspicious messages from multiple devices were aggregated while adhering to privacy and anonymisation requirements (federated learning). This enabled the formation of updated training datasets and prevents changes in the statistical properties of the data over time, which could otherwise lead to model performance degradation. The use of continuous learning mechanisms and model stability control techniques (such as change buffers and validation set checks) helped avoid overfitting to short-term anomalies and maintains the system's generalisation capability over the long term.

D. Honeycutt *et al*. (2020) investigated the impact of interactive user feedback in human-in-the-loop machine learning systems. The authors conducted an experiment using a simulated image object recognition system to assess how the ability to provide feedback affects users' trust in the system and their perception of model accuracy. Although interactive feedback could potentially improve system performance, the results showed a decrease in both trust and perceived accuracy among users who participated in providing feedback. This suggested that the mere opportunity for users to intervene in the system's operation can have unexpected psychological effects, influencing technology acceptance. Comparing these results with the present study, where integration of user feedback and system logs was used to update models and personalise spam filtering, it is important to consider potential challenges in maintaining user trust. Although the proposed approach enhances system adaptability and its effectiveness in detecting new threats, it highlighted the importance of balancing automation with user involvement to avoid undermining trust or diminishing the user experience. Therefore, improving the models should take into account not only technical aspects but also interface design and interaction methods with end users.

Experimental results on the publicly available SMS Spam Collection Dataset confirmed the high accuracy and adaptability of the proposed system. The use of a comprehensive sampling structure for retraining, which accounts for the weights of different message types and the reliability of feedback, enabled an effective balance between model stability and flexibility. In particular, increased focus on misclassified cases and messages with low confidence helped reduce errors and improved overall classification quality without the risk of overfitting to random noise in the data. The conclusions align with the findings of the aforementioned studies: hybrid architectures indeed enhance both accuracy and adaptability, while the incorporation of privacy and robustness mechanisms represents a contemporary trend. On the other hand, the approach differed by integrating stability control mechanisms, weighted sampling, and a feedback loop, which collectively increased the reliability of the system in real-time operation.

All in all, the proposed hybrid SMS spam detection system reflects the current global trend towards developing adaptive, scalable, and personalised solutions for real-time information security. It successfully combines the advantages of local processing with cloud technologies, providing high accuracy, stability, and the ability to respond rapidly to emerging threats in a dynamic information environment. An important advantage of the proposed architecture was that cloud-level models are activated only when the local model exhibits low confidence or when new suspicious patterns typical of spam are detected. This conditional delegation strategy provided the ability to optimise the use of computational resources and reduce server load, while at the same time maintaining high classification accuracy. Thus, the selected set of models implemented the

principle of functional distribution of computations: simple and fast algorithms provide preliminary filtering on the user's device, while more complex models in the cloud performed deeper analysis only when required. This approach combined the advantages of speed, adaptability, and scalability, ensuring the reliable operation of the system in real time under the conditions of a dynamically changing spam environment.

CONCLUSIONS

The conducted study demonstrated the effectiveness of a hybrid SMS spam detection system that combines lightweight, local models for rapid filtering with deep learning solutions deployed in the cloud. This architecture ensures high classification accuracy while minimising computational costs, making it particularly suitable for devices with limited processing power. The integration of fast local evaluation with more resource-intensive cloud-based analysis reduces latency and optimises overall performance. Dynamic updating of knowledge models, informed by both user feedback and system logs, enables the system to adapt promptly to emerging spam patterns. This adaptability ensures robust generalisation, even as the statistical properties of the data evolve over time.

The system's performance was evaluated using the publicly available SMS Spam Collection Dataset and standard classification metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. Comparative analysis showed that deep learning models – particularly LSTM – significantly outperformed traditional classifiers across all metrics, achieving an accuracy and AUC of 0.98. These results highlight the model's strong ability to distinguish between spam and legitimate content, even in complex scenarios. Among the tested methods (Naïve Bayes, Logistic Regression, Random Forest), LSTM achieved the highest performance and was utilised at the cloud level of the system. While classical models such as Naïve Bayes demonstrated acceptable results (accuracy of 0.92 and F1-score of 0.86), their inability to capture sequential dependencies between words limited their effectiveness in detecting more sophisticated or hidden patterns. This finding underscores the necessity of leveraging advanced neural architectures within cloud-based components.

Future research should focus on enhancing adaptive learning mechanisms through the development of more flexible feedback processing strategies, improving accuracy with limited data, and reducing update latency in the cloud. Moreover, the ethical use of personal data and the protection of user privacy during large-scale SMS processing in cloud environments remain critical areas for ongoing investigation.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abid, M.A., Ullah, S., Siddique, M.A., Siddique, M.A., Mushtaq, M.F., Alijedaani, W., & Rustam, F. (2022). Spam SMS filtering based on text features and supervised machine learning techniques. *Multimedia Tools and Applications*, 81, 39853-39871. doi: 10.1007/s11042-022-12991-0.

[2] Ahmadi, M., Khajavi, M., Varmaghani, A., Ala, A., Danesh, K., & Javaheri, D. (2025). Leveraging large language models for cybersecurity: Enhancing SMS spam detection with robust and context-aware text classification. *ArXiv*. doi: 10.48550/arXiv.2502.11014.

[3] Al Maruf, A., Al Numan, A., Haque, M.M., Jidney, T.T., & Aung, Z. (2023). Ensemble approach to classify spam SMS from Bengali text. In M. Singh, V. Tyagi, P. Gupta, J. Flusser & T. Ören (Eds.), *Advances in computing and data sciences. ICACDS 2023* (pp. 440-453). Cham: Springer. doi: 10.1007/978-3-031-37940-6_36.

[4] Almeida, T. & Hidalgo, J. (2011). SMS spam collection. *UCI Machine Learning Repository*. doi: 10.24432/C5CC84.

[5] Al-Zebari, A., Barwary, M., Omar, N., Zebari, N.A., & Zebari, D.A. (2025). Deep learning hybrid approach for accurate SMS spam identification. *Journal of Information Systems Engineering and Management*, 10(10s). doi: 10.52783/jisem.v10i10s.1426.

[6] Baaqeel, H., & Zagrouba, R. (2020). Hybrid SMS spam filtering system using machine learning techniques. In *2020 21st international Arab conference on information technology (ACIT)* (pp. 1-8). Giza: IEEE. doi: 10.1109/ACIT50332.2020.9300071.

[7] Bäckman, D. (2019). *Evaluation of machine learning algorithms for SMS spam filtering*. (Bachelor's thesis, Umeå University, Umeå, Switzerland).

[8] Boyko, N., & Kovalchuk, R. (2023). Data update algorithms in the machine learning system. *Computer Systems and Information Technologies*, 1, 6-13. doi: 10.31891/csit-2023-1-1.

[9] Gadde, S., Lakshmanarao, A., & Satyanarayana, S. (2021). SMS spam detection using machine learning and deep learning techniques. In *2021 7th international conference on advanced computing and communication systems (ICACCS)* (pp. 358-362). Coimbatore: IEEE. doi: 10.1109/ICACCS51430.2021.9441783.

[10] Gomaa, W.H. (2020). The impact of deep learning techniques on SMS spam filtering. *International Journal of Advanced Computer Science and Applications*, 11(1), 544-549. doi: 10.14569/IJACSA.2020.0110167.

[11] Honeycutt, D.R., Nourani, M., & Ragan, E.D. (2020). Soliciting human-in-the-loop user feedback for interactive machine learning reduces user trust and impressions of model accuracy. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 8(1), 63-72. doi: 10.1609/hcomp.v8i1.7464.

[12] Hossain, S.M.M., Sumon, J.A., Sen, A., Alam, M.I., Kamal, K.M.A., Alqahtani, H., & Sarker, I.H. (2022). Spam filtering of mobile SMS using CNN-LSTM based deep learning model. In *Hybrid intelligent systems* (pp. 106-116). Cham: Springer. doi: 10.1007/978-3-030-96305-7_10.

[13] Kalyani, V.V., Rama Sundari, M.V., Neelima, S., Satya Prasad, P.S., PattabhıRama Mohan, P., & Lakshmanarao, A. (2024). SMS spam detection using NLP and deep learning recurrent neural network variants. In *2024 international conference on cognitive robotics and intelligent systems (ICC – ROBINS)* (pp. 92-96). Coimbatore: IEEE. doi: 10.1109/ICC-ROBINS60238.2024.10533895.

[14] Li, Y., Zhang, R., Rong, W., & Mi, X. (2024). SpamDam: Towards privacy-preserving and adversary-resistant SMS spam detection. *ArXiv*. doi: 10.48550/arXiv.2404.09481.

[15] Mohammed, C.N., & Ahmed, A.M. (2024). A semantic-based model with a hybrid feature engineering process for accurate spam detection. *Journal of Electrical Systems and Information Technology*, 11, article number 26. doi: 10.1186/s43067-024-00151-3.

[16] Molina-Coronado, B., Mori, U., Mendiburu, A., & Miguel-Alonso, J. (2023). Efficient concept drift handling for batch Android malware detection models. *ArXiv*. doi: 10.48550/arXiv.2309.09807.

[17] Oyeyemi, D.A., & Ojo, A.K. (2024). SMS spam detection and classification to combat abuse in telephone networks using natural language processing. *Journal of Advances in Mathematics and Computer Science*, 38(10), 144-156. doi: 10.9734/jamcs/2023/v38i101832.

[18] Prashob, J., & Yerima, S.Y. (2022). A comparative study of word embedding techniques for SMS spam detection. In *14th IEEE international conference on computational intelligence and communication networks (CICN 2022)* (pp. 149-155). Al-Khobar: IEEE. doi: 10.1109/CICN56167.2022.10008245.

[19] Rojas-Galeano, S. (2021). Using BERT encoding to tackle the Mad-lib attack in SMS spam detection. *ArXiv*. doi: 10.48550/arXiv.2107.06400.

[20] Salman, M., Ikram, M., & Kaafar, M.A. (2024). Investigating evasive techniques in SMS spam filtering: A comparative analysis of machine learning models. *IEEE Access*, 12, 24306-24324. doi: 10.1109/ACCESS.2024.3364671.

Vats, S., Shastri, S., & Mehta, S. (2024). Federated learning for SMS spam detection: A privacy-focused approach. *2024 15th international conference on computing communication and networking technologies (ICCCNT)* (pp. 1-5). Kamand: IEEE. doi: 10.1109/ICCCNT61001.2024.10724879.

Адаптивна гібридна система виявлення SMS-спаму з самонавчанням за фідбеком користувача

Ігор Лях

Доктор технічних наук, професор
Державний вищий навчальний заклад «Ужгородський національний університет»
88000, пл. Народна, 3, м. Ужгород, Україна
https://orcid.org/0000-0001-5417-9403

Андрій Чорній

Кандидат технічних наук, доцент
Черкаський державний технологічний університет
18006, б-р Шевченка, 460, м. Черкаси, Україна
https://orcid.org/0000-0002-0989-711

Оксана Лутак

Магістрантка
Державний вищий навчальний заклад «Ужгородський національний університет»
88000, пл. Народна, 3, м. Ужгород, Україна
https://orcid.org/0009-0002-0798-6928

Мар'ян Ценкнер

Магістрант
Державний вищий навчальний заклад «Ужгородський національний університет»
88000, пл. Народна, 3, м. Ужгород, Україна
https://orcid.org/0009-0003-2752-7398

Анотація. У статті представлено комплексний підхід до виявлення SMS-спаму на основі гібридної архітектури, яка поєднує локальні алгоритми обробки повідомлень із високопродуктивними хмарними моделями глибокого навчання. Такий підхід дозволяє досягти балансу між точністю та конфіденційністю обробки вхідних повідомлень. Метою дослідження було створення інтелектуальної гібридної системи виявлення SMS-спаму, яка забезпечувала б високу точність класифікації, підтримку актуальності знань, персоналізацію для користувачів та здатність адаптуватися до нових шаблонів атак. Для досягнення мети дослідження було застосовано комплексний аналітичний підхід, що поєднував детальний огляд наукової літератури з питань виявлення спаму в SMS-повідомленнях, включаючи машинне навчання, нейронні мережі та гібридні методи, з емпіричним аналізом. Для реалізації класичних моделей машинного навчання (Naïve Bayes, Logistic Regression, Random Forest) використовувалися стандартні бібліотеки машинного навчання, а для глибокого навчання – фреймворки, що підтримують рекурентні нейронні мережі, зокрема Long Short-Term Memory та трансформерні архітектури. Тестування системи на відкритому датасеті SMS Spam Collection з використанням метрик Accuracy (до 0,98), F1-score (до 0,95) та ROC-AUC (до 0,98). Додатково було реалізовано механізм динамічного оновлення знань через зворотний зв'язок користувача та запропоновано вагову систему оцінки достовірності фідбеку. У ході дослідження було розроблено багаторівневу систему, що виконувала початкову класифікацію на пристрої користувача з можливістю делегування обробки хмарному модулю у випадках невизначеності. У порівнянні з базовими підходами, гібридна архітектура продемонструвала покращення точності класифікації, зниження кількості хибнопозитивних і хибнонегативних спрацьовувань, а також підвищену адаптивність до змін у структурі spam-повідомлень. Агрегація підозрілих повідомлень у хмарі забезпечувала ефективне донавчання моделей у випадках концептуального зсуву. Практична цінність результатів полягає в можливості інтеграції розробленої системи для мобільних платформ, а також у корпоративні засоби інформаційної безпеки з метою фільтрації SMS-контенту та захисту кінцевих користувачів від соціальної інженерії

Ключові слова: обробка природної мови; довга короткочасна пам'ять; архітектура спаму; повідомлення; метрика

ВІСНИК
Черкаського державного технологічного університету

Науковий збірник

Том 30, № 3. 2025

Заснований у 1996 р. Виходить чотири рази на рік

Оригінал-макет видання виготовлено у редакційно-видавничому відділі
Черкаського державного технологічного університету

Відповідальний редактор:
А. Лавданський

Підписано до друку 15 вересня 2025 р.
Формат 60*84/8
Умов. друк. арк. 15,6
Наклад 50 прим.

Адреса видавництва:
Черкаський державний технологічний університет
18006, бульв. Шевченка, 460, м. Черкаси, Україна
Тел.: +380638013283
E-mail: info@bulletin-chstu.com.ua
https://bulletin-chstu.com.ua/uk

BULLETIN
of Cherkasy State Technological University

Scientific Collection

Volume 30, No. 3. 2025

Founded in 1996. Published four times per year

The original layout of the publication is made in the editorial and publishing department
of Cherkasy State Technological University

Managing editor:
A. Lavdanskyi

Signed for print of September 15, 2025.
Format 60*84/8
Conventional printed pages 15.6
Circulation 50 copies

Editors Office Address:

Cherkasy State Technological University
18006, 460 Shevchenko Blvd., Cherkasy, Ukraine
Tel.: +380638013283
E-mail: info@bulletin-chstu.com.ua
https://bulletin-chstu.com.ua/en