The 15th Conference of PhD Students in Computer Science

Volume of short papers

CS^2

Organized by the Institute of Informatics of the University of Szeged

July 1 – 3, 2026
Szeged, Hungary

Scientific Committee:

Tibor Csendes (co-chair, University of Szeged)
György Vaszil (co-chair, University of Debrecen)
János Balogh (member, University of Szeged)
József Békési (member, University of Szeged)
András Benczúr (member, Eötvös Loránd University)
Árpád Beszédes (member, University of Szeged)
Vilmos Bilicki (member, University of Szeged)
Erzsébet Csuhaj-Varjú (member, Eötvös Loránd University)
József Dombi (member, University of Szeged)
József Dániel Dombi (member, University of Szeged)
Richárd Farkas (member, University of Szeged)
István Fazekas (member, University of Debrecen)
Hartung Ferenc (member, University of Pannonia)
Zsolt Gazdag (member, University of Szeged)
Boglárka Gazdag-Tóth (member, University of Szeged)
Katalin Hangos (member, SZTAKI)
Péter Hegedűs (member, University of Szeged)
Zoltán Horváth (member, Eötvös Loránd University)
Márk Jelasity (member, University of Szeged)
Zoltán Keresztes (member, University of Szeged)
Attila Kertész (member, University of Szeged)
Ákos Kiss (member, University of Szeged)
László T. Kóczy (member, Budapest University of Technology and Economics)
Andrea Kő (member, Corvinus University of Budapest)
András London (member, University of Szeged)
Gyöngyvér Márton (member, Sapientia College of Theology of Religious Orders)
Róbert Mingesz (member, University of Szeged)
Zoltán Nagy (member, Pázmány Péter Catholic University)
Antal Nagy (member, University of Szeged)
László Nyúl (member, University of Szeged)
Ákos Odry (member, University of Szeged, University of Dunaújváros)
Kálmán Palágyi (member, University of Szeged)
Attila Pethő (member, University of Debrecen)
Gábor Szederkényi (member, Pázmány Péter Catholic University)
Gergely Vadai (member, University of Szeged)
János Végh (member, University of Debrecen)
László Vidács (member, University of Szeged)
Zsolt Vörösházi (member, University of Pannonia)

Organizing Committee:

Zoltán Kincses (chair),
Balázs Bánhelyi,
Tamás Gergely,
Judit Jász,
László Schäffer
Eszter Kun

Address of the Organizing Committee

c/o. Zoltán Kincses
University of Szeged, Institute of Informatics
H-6701 Szeged, P.O. Box 652, Hungary
Phone: +36 62 543 281
E-mail: cscs@inf.u-szeged.hu
URL: http://www.inf.u-szeged.hu/~cscs/

Sponsors

University of Szeged,
Institute of Informatics

3i Fejlesztő és Szolgáltató Kft.

Preface

This conference is the 15th in a series. The organizers aimed to bring together PhD students working on any field of computer science and its applications to help them publish one of their first papers and provide an opportunity to hold a scientific talk. To our knowledge, this is one of the few such conferences. The aims of the scientific meeting were determined at the council meeting of the Hungarian PhD Schools in Informatics: it should:

- provide a forum for PhD students in computer science and engineering to discuss their ideas and research results,
- give a possibility to have constructive criticism before they present the results at professional conferences,
- promote the publication of their results in the form of fully refereed journal articles and, finally,
- promote hopeful fruitful research collaboration among participants.

The papers emerging from the presented talks will be invited to be considered for full paper publication in the Acta Cybernetica journal.

Szeged, July 2026

Zoltán Kincses
Balázs Bánhelyi
Tamás Gergely
Judit Jász
László Schäffer
Eszter Kun

Contents

Program

Wednesday, July 1

09:00 – 09:40	Registration
09:40 – 09:50	Opening
09:50 – 10:30	Talks – Analisys Methods (2x20 min.)
10:30 – 11:00	Coffee Break
11:00 – 12:00	Plenary Talk
12:00 – 14:00	Lunch Break
14:00 – 15:40	Talks – Artifical Intelligence 1 (5x20 min.)
15:40 – 16:10	Coffee Break
16:10 – 17:50	Talks – Artifical Intelligence 2 (5x20 min.)
18:15 – 19:00	Sightseeing Train
19:00 –	Welcome Party

Thursday, July 2

08:50 – 10:30	Talks – Computer Graphics and Image Processing (5x20 min.)	
10:30 – 11:00	Coffee Break	
11:00 – 12:00	Plenary Talk	
12:00 – 14:00	Lunch Break	
14:00 – 15:20	*14:00 – 15:00* Talks – Application (3x20 min.)	*14:00 – 15:20* Talks – Graphs (4x20 min.)
16:20 – 19:00	Organ recital and Dome Tower visit	
19:00 –	Pizza & Streetfood Party	

Friday, July 3

08:50 – 10:30	Talks – Security and Vulnerability (5x20 min.)
10:30 – 11:00	Coffee Break
11:00 – 12:00	Plenary Talk
12:00 – 14:00	Lunch Break
14:00 – 15:40	Talks – Computation (5x20 min.)

Detailed program

Wednesday, July 1

09:00	**Registration**
09:40	**Opening**
Session 1	**Analysis Methods** - Session chair: István Siket
09:50	Aleksandr Samedov and Tamás Kozsik: *Experiments on the Energy Efficiency of Static Analysis*
10:10	Abbas Alharan, István Kósa, Brigitta Szálka and István Vassányi: *Mining dietary harmony rules from user logs for automated meal plan generation*
10:30	Coffee Break
11:00	**Plenary Talk** - Gergely Ambrus: *Extreme Tales on Unit Distances*
12:00	Group photo before lunch
12:10	**Lunch Break**
Session 2	**Artifical Intelligence 1** - Session chair: Gábor Gosztolya
14:00	Szolnoki Norbert Sándor and Gábor Antal: *Fine-Tuning CodeBERT-Java for Method-Level Bug Detection on Defects4J*
14:20	Nándor Ákos Vincze and Balázs Bánhelyi: *Reliable Parking Trajectory Prediction with Verifiable Neural Networks*
14:40	Zahra Delavari: *AI Agent-based Approach for Enhancing Urban Mobility Applications*
15:00	Balázs Ádám Toldi, András Földvári, András Pataricza and Zsigmond Pap: *Deterministic Diagnostics for Highly Configurable 5G Networks Using Inductive Learning*
15:20	Vivien Vörös, Gábor Antal and Péter Hegedűs: *Comparing Large Language Models for Automated Vulnerability Repair under a Fixed Prompting Strategy*
15:40	Coffee Break

Session 3	**Artifical Intelligence 2** - Session chair: Gábor Berend
16:10	Moenes Benaissa and László József Kovács: *Taxonomy and Optimization of Kolmogorov-Arnold Networks vs. Multi-Layer Perceptrons for Explainable AI*
16:30	Hiba Adil Al-Kharsan and Robert Rajkó: *Diffusion-Based Feature Denoising with NNMF for Robust handwritten digit multi-class classification*
16:50	Krisztián Pomázi and Bertalan Forstner: *Meteorology-Driven Live Fuel Moisture Content Estimation at Global Scale: A Machine Learning Empirical Study Without Satellite Imagery*
17:10	István Kolláth and Gábor Antal: *Evaluating LLMs for SAST Warning Repair: An Empirical Study on VUL4J*
17:30	Ádám Domonkos and Péter Ekler: *Passive Detection of Forward Error Correction Schemes from Network Traffic*
18:15	Sightseeing Train
19:00	Welcome Party

Thursday, July 2

Session 4	**Computer Graphics and Image Processing** - Session chair: Péter Balázs
08:50	Viktor Vad and Gábor Valasek: *Point-Curve Minimal Distance via Higher Order Taylor Estimation of Footpoint Mapping*
09:10	Hanan Namrouti, Cecília Sik-Lányi and Tibor Guzsvinecz: *Cross-Cultural Color Fidelity and Realism: A Comparative Perceptual Study across Real, Unreal Engine, and Unity Lighting Environments*
09:30	Noel Nagy and Kálmán Palágyi: *Fully Parallel 3D Curve-Thinning on the FCC grid with Endpoint Re-Checking*
09:50	Gergő Papp, Emil Világos, Krisztián Szekeres and Ellák Somfai: *Athlete Tracking and Identification in Water Polo*
10:10	Anna Lili Horváth, Gábor Valasek and Csongor Csanád Karikó: *Spline Proxies for Planar Level Set Distance Computations*
10:30	Coffee Break
11:00	**Plenary Talk** - Zsolt Gazdag: *Context Matters: How Context Sensitivity Enhances Computational Power in Membrane Systems*
12:00	Group photo before lunch
12:10	**Lunch** - Cheers Bar
Session 5	**Application (Parallel)** - Session chair: Gergely Vadai
14:00	Ádám Brand and Attila Magyar: *Fuzzy Rule-Based Smart Street Lighting for Energy-Efficient Urban Illumination*
14:20	Etele Balogh and András Kicsi: *Prediction of 90-day mRS values of stroke patients based on textual care documentation*
14:40	Róbert Nagy: *Implementation and computational verification of digital signal processing pipelines for temporal light artifact measurements*
Session 6	**Graphs (Parallel)** - Session chair: Tibor Csendes
14:00	László Gyarmati: *Optimal Object Arrangements for Star Graph Comparison Structures in the Bradley-Terry model*
14:20	Dávid Maliga and Levente Buttyan: *Design and analysis of incremental clustering algorithms for large dynamic similarity graphs*
14:40	Péter Hajnal, Edit Pengő and István Siket: *Technical Challenges in Java Symbolic Execution: Control Flow Graphs and Exception Handling Dynamics*
15:00	Dániel Kovács and Milán Mondok: *Reusing Intermediate Model Checking Results With Implicit Abstraction*
16:20	Organ recital and Dome Tower visit
19:00	Pizza & Streetfood Party

Friday, July 3

Session 7	**Security and Vulnerability** - Session chair: Ákos Kiss
08:50	Csaba Nagy, Andrea Huszti and Norbert Oláh: *Blockchain-based RAG Architecture for Vulnerability Assessment*
09:10	József Sándor, Bence Kovács and Levente Buttyán: *Towards Automated Generation of Adversarial Malware Samples with a Genetic Algorithm*
09:30	Yusufbek Sulaymonov and Veronika Szűcs: *Keystroke-Based Behavioral Liveness Detection for Adaptive Multi-Factor Authentication*
09:50	Sami Abdel-Fattah and Judit Jász: *Enhancing LLM-Based Vulnerability Analysis Using Runtime-Aware Context: An Empirical Study on the Vul4J Dataset*
10:10	Yusufbek Sulaymonov and Veronika Szűcs: *A Multi-Criteria Scoring Model for Evaluating and Selecting Multi-Factor Authentication Mechanisms*
10:30	Coffee Break
11:00	**Plenary Talk** - Róbert Trényi: *Iterative Optimization in Quantum Metrology and Entanglement Theory Using Semidefinite Programming*
12:00	Group photo before lunch
12:10	**Lunch Break**
Session 8	**Computation** - Session chair: András London
14:00	Roland Kotroczó and János Márk Szalai-Gindl: *Data Ordering for Split-Based M-tree Construction*
14:20	Bálint Tóth and Zoltán Juhász: *A High-Performance GPU-accelerated Parallel 2D Multigrid Poisson Solver*
14:40	Bertalan Kovács, Botond Bertók and Márton Frits: *Hybrid bitset-based data types for optimizing runtimes of set operations*
15:00	Zoltán Kégli, Tamás Kozsik and Péter Rakyta: *Bounded Exact Channel-Native Fusion of Local Noisy Motifs in Density-Matrix Simulation*
15:20	Pál Mihály: *From Centralized to Distributed Digital Twins: A Review*

Extreme Tales on Unit Distances

Gergely Ambruzs
University of Szeged, Szeged, Hungary

I will tell the story of two of Erdős's favourite problems that culminate in modern and unforeseen twists.

Context Matters: How Context Sensitivity Enhances Computational Power in Membrane Systems

Zsolt Gazdag
University of Szeged, Szeged, Hungary

It is well known that context sensitivity plays an important role in the theory of formal languages. In my presentation, I discuss how different forms of context sensitivity influence the computational power of a bio-inspired computing model known as P systems with active membranes [3]. These abstract models are parallel computing devices inspired by the structure and functioning of living cells. Such a system consists of hierarchically organized regions, each bounded by membranes that possess specific polarizations. The regions contain multisets of objects. Both membranes and objects are manipulated by means of developmental rules, including evolution rules (objects can be transformed into other objects), communication rules (objects can be transported across membranes), division rules (objects can divide elementary membranes, duplicating the contents of the original membrane), and dissolution rules (objects can dissolve membranes).

All of these rules are initiated by a single object; hence, context sensitivity is not explicitly incorporated into the rule set itself. However, context sensitivity arises implicitly through the capacity of objects to obtain information about the presence of other objects within the same membrane. The most direct mechanism to achieve this is the use of polarizations. Alternatively, an object may dissolve a membrane, causing all the objects contained in that membrane to be moved into the parent membrane. Any object that thus ends up in the parent membrane receives information that the object that dissolved the membrane was present alongside it in the now-dissolved membrane.

It is known that polynomial-time P systems with active membranes characterize the complexity class $\mathrm{P}^{\#\mathrm{P}}$, that is, the class of decision problems that can be solved in polynomial time by Turing machines equipped with #P oracles—and this holds even when dissolution rules are not allowed [2]. On the other hand, if membrane polarizations are omitted, it is conjectured that these P systems can efficiently solve only problems in P. The P lower bound in this conjecture is straightforward, and notably, it still applies even in the restricted case when only dissolution rules are permitted [1]. In my presentation, I will explore this topic in more detail and discuss some methods described in the literature for replacing the context sensitivity provided by membrane polarizations with other mechanisms.

References

[1] Gazdag, Z., Hajagos, K.: *On the power of membrane dissolution in polarizationless P systems with active membranes.* Natural Computing **22**, 95–104 (2022)

[2] Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: *Simulating elementary active membranes, with an application to the P conjecture.* In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron (Eds.) Membrane Computing – 15th International Conference, CMC15, LNCS vol. 8961, 284–299 (2014)

[3] Păun, Gh.: *P systems with active membranes: attacking NP-complete problems.* Journal of Automata, Languages and Combinatorics **6**(1), 75–90 (2001)

Iterative Optimization in Quantum Metrology and Entanglement Theory Using Semidefinite Programming

Róbert Trényi
University of Szeged, Szeged, Hungary

Quantum metrology [1] is about measuring physical parameters with precision beyond classical limits by exploiting quantum effects. The metrological performance of a quantum state is measured by how much it can outperform all separable states in a metrological task. We present efficient optimization techniques to maximize this performance by searching for the optimal local Hamiltonian generating the unitary dynamics for a given bipartite initial state [2]. We show that this is equivalent to maximizing the Quantum Fisher Information over a specific set of local Hamiltonians. This task is highly non-trivial, as it involves maximizing a convex function over a convex set. We reformulate the problem in a bilinear way and optimize it using an iterative see-saw method, where each optimization step is solved via semidefinite programming. This way we obtain a lower bound to the maximally achievable performance. We also apply relaxation methods that give upper bounds for the optimum.

We further show that the same optimization framework can be adapted to problems in entanglement theory, such as identifying bound entangled states that maximally violate the Computable Cross Norm-Realignment criterion. Finally, we provide examples where two copies of a quantum state outperform a single copy, demonstrating metrological activation [3] for certain small systems.

References

[1] L. Pezzé, A. Smerzi, M. K. Oberthaler, R. Schmied, P. Treutlein, Quantum metrology with nonclassical states of atomic ensembles, *Rev. Mod. Phys.* **90** (2018), 035005.

[2] Á. Lukács, R. Trényi, T. Vértesi, G. Tóth, Iterative optimization in quantum metrology and entanglement theory using semidefinite programming, *Quantum Sci. Technol.* **11** (2026), 015042.

[3] R. Trényi, Á. Lukács, P. Horodecki, R. Horodecki, T. Vértesi, G. Tóth, Activation of metrologically useful genuine multipartite entanglement, *New J. Phys.* **26** (2024), 023034.

Experiments on the Energy Efficiency of Static Analysis

Samedov Aleksandr, Kozsik Tamas

Abstract: This paper investigates the energy efficiency of a parallel static analyzer designed to verify the algorithmic complexity of Java methods. Using Abstract Syntax Tree (AST) pattern-matching, we conducted a dual-level energy evaluation: macro-profiling via JMH (Intel RAPL) and micro-profiling via JoularJX. Our findings confirm the "race-to-sleep" principle: parallelizing the analysis to an optimal 6-thread configuration reduced execution time by approximately 66% and total energy consumption by 13% compared to a single-threaded baseline. However, increasing the thread count to 12 resulted in a "concurrency collapse", where energy consumption peaked at 27.03 J due to I/O contention and Garbage Collection overhead. Micro-profiling revealed that AST generation via JavaParser is the primary bottleneck, accounting for ∼85–89% of energy, while core complexity detection logic consumes less than 1%. We conclude that green optimization should prioritize AST caching over algorithmic refinement.

Keywords: Green Computing, Static Analysis, Energy Efficiency, Parallel Processing, JMH, JoularJX, RAPL

1 Introduction

Static analyzers are essential in modern CI/CD pipelines but consume significant computational resources. In Green Computing, understanding their energy behavior is vital as these tools run frequently during development [2]. This work evaluates the Java-based parallel static complexity analyzer introduced by Samedov and Porkoláb [1], which verifies algorithmic complexity from AST patterns. The main contribution of this paper is an energy measurement methodology which combines macro-profiling with JMH [3] and RAPL [4], and micro-profiling with JoularJX [5] to identify software-level bottlenecks. Another contribution is the analysis of the energy consumption properties of a static analyzer tool with respect to parallel execution.

2 Methodology

2.1 Reproducibility: Hardware/Software and Workload

Measurements were collected on a workstation with an Intel Core i7–10750H CPU (6 cores/12 threads) and 32 GB RAM, running Ubuntu 22.04. The custom parallel analyzer tool [1] utilizes the JavaParser library [6] for AST processing. The analyzed workload consists of a fixed corpus of 1,500 Java source files from the Spring Boot Core framework [10], averaging 300 LOC/file with standard deviation, and containing various algorithmic patterns to exercise different complexity detection rules.

2.2 System-Level Macro-Profiling (JMH)

We used JMH to orchestrate benchmarks, mitigating JVM warm-up issues like JIT compilation and garbage collection [7]. Total system energy was captured via Intel RAPL hardware counters [8]. We tested scalability from 1 to 12 threads to identify the efficiency "sweet spot".

2.3 Method-Level Micro-Profiling (JoularJX)

JoularJX, a Java agent using bytecode instrumentation and stack sampling, was used for method-level energy attribution [5]. While instrumentation introduces absolute overhead, it allows us to calculate the relative percentage of energy consumed by specific methods, creating an energy call-graph [7, 3].

3 Experimental Results

There were two main categories of results: system-level macro-profiling and method-level micro-profiling. Both included 33 iterations per thread configuration to ensure statistical significance and mitigate outliers.

3.1 System-Level Energy and Scalability

Table 1 displays the scalability results. The analyzer follows a "Race-to-Sleep" pattern [9] up to 6 threads, where parallelization completes the workload faster, allowing the CPU to return to low-power idle states sooner.

Table 1: System-Level Performance and Energy (JMH + RAPL)

Configuration	**Avg Time (ms)**	**Speedup**	**Energy (J/op)**
1 Thread (Baseline)	1424.64	1.00x	23.50
6 Threads (Optimal)	482.59	2.95x	20.48
12 Threads (Overload)	653.31	2.18x	27.03

3.2 Method-Level Energy Attribution

While macro-profiling captures system-wide trends, micro-profiling via JoularJX isolates the internal method-level energy distribution. Table 2 outlines the relative energy consumption shifts for the application logic, underlying library parsing, system I/O, and memory management subsystems.

Table 2: JoularJX Energy Distribution Shift Across Thread Counts (Joules)

Subsystem / Key Operation	**1 Thread**	**6 Threads**	**12 Threads**
AST Parsing (`parseFile`)	1303.65	1479.97	1403.57
File Discovery (`collectJavaFiles`)	50.65	112.54	277.89
System-Level I/O (`read` + OS calls)	Negligible	272.73	533.55
Memory / GC (`waitForReference`)	Negligible	Negligible	120.36
Core Pattern Matching Logic	< 1.00	< 1.00	< 1.00

The micro-profiling results reveal that Abstract Syntax Tree (AST) consistently dominates the internal application energy profile across all thread settings. However, as concurrency scales, severe energy inflation emerges within the file discovery, character-stream reading, and native operating system subsystems. Conversely, the complexity characteristics of the analyzed corpus draw negligible power (less than 1 Joule) regardless of the thread configuration.

4 Discussion

We applied our energy measurement methodology to a particular application, namely a static analysis tool. The current approach is tied to the Java language because it relies on JavaParser and JoularJX, and to the Linux operating system because it depends on Intel RAPL-based measurements and the Ubuntu runtime environment. However, these limitations can be mitigated by choosing other measurement tools for different contexts.

In this particular experiment, the macro- and micro-profiling results collectively highlight a definitive performance "Sweet Spot" at 6 threads. Up to this configuration, parallelization achieves an efficient "Race-to-Sleep" hardware phenomenon: reducing execution time by nearly

3x allows the processor to quickly drop back into a low-power idle state, minimizing total Joules per operation.

Beyond 6 threads, the micro-profiling data in Table 2 provides empirical evidence of a "Concurrency Collapse" driven by two distinct architectural limitations:

- **The I/O Wall:** At 12 threads, system-level I/O consumption skyrockets to 533.55 Joules, which is triggered by JavaParser's character-stream reader and native UNIX storage subroutines. Multiple threads flood the OS with simultaneous read requests faster than the storage hardware can fulfill them, CPU-bounding an inherently I/O-bound operation and causing threads to spin idly while waiting for file locks.
- **The Memory Wall:** Pushing the workload to 12 threads introduces an intense object allocation rate on the JVM heap, as multiple concurrent parsers instantiate millions of transient AST nodes. This allocation pressure forces frequent Garbage Collection pauses, causing the deep internal JVM routine to consume over 75 Joules in wasted energy overhead.

Additionally, directory tree traversal routines experience a 5.5x energy penalty when running on 12 threads compared to 1 thread, verifying that parallel file discovery incurs severe synchronization and locking overhead.

These findings indicate that future optimization efforts for static analysis pipelines should pivot away from code logic refinements. Instead, substantial energy savings can be generalized to other AST-heavy architectures by introducing aggressive file-level disk caching and incremental parsing mechanisms to circumvent the expensive tokenization and parsing stages entirely.

5 Conclusion

Combining macro-level and micro-level benchmarking provides comprehensive information about the energy consumption behavior of software. This study proves that for static analysis, energy efficiency is tied to I/O and parsing rather than algorithmic complexity. Optimal concurrency (6 threads) provides significant gains, but over-subscription leads to systemic "traffic jams" in memory and I/O. Future optimization should prioritize AST caching and incremental parsing to bypass the expensive JavaParser engine during sequential runs.

Acknowledgements

This work was supported by project no. TKP2021-NVA-29 under the Ministry of Innovation and Technology of Hungary from the National Research, Development, and Innovation Fund and financed under the TKP2021-NVA funding scheme.

References

[1] A. Samedov and Z. Porkolab, Annotation-Based Static Verification of Algorithmic Complexity in Java *Proceedings of the Twelfth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA 2025)*, Maribor, Slovenia, September 10–12, 2025. *CEUR Workshop Proceedings*, Vol. 4077, paper 16, CEUR-WS.org.

[2] G. Pinto and F. Castor. Energy efficiency: a new concern for application software developers, *Communications of the ACM*, 60(12):68–75, 2017.

[3] A. Shipilev. JMH: Java Microbenchmark Harness, *OpenJDK*, Available: https://openjdk.java.net/projects/code-tools/jmh/, 2014.

[4] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. Measuring energy and power with PAPI, *41st International Conference on Parallel Processing Workshops (ICPPW)*, 262–268, IEEE, 2012.

[5] A. Noureddine. JoularJX: Software power profiling for Java. *GitHub Repository*, Available: https://github.com/joular/joularjx, 2023.

[6] JavaParser Contributors. JavaParser: Analyze, transform and generate your Java codebase, *GitHub Repository*, Available: https://github.com/javaparser/javaparser, 2023.

[7] A. Georges, D. Buytaert, and L. Eeckhout. Statistically rigorous java performance evaluation, *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems and applications (OOPSLA)*, 57–76, 2007.

[8] Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide (Chapter 14.9–Intel RAPL), 2023.

[9] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power (Race to Sleep/Halt concepts), *Proceedings of the 28th Annual International Symposium on Computer Architecture*, 240–251, 2001.

[10] VMware Tanzu. Spring Boot Framework Reference Documentation, Available: https://github.com/spring-projects/spring-boot, 2026.

Mining dietary harmony rules from user logs for automated meal plan generation

Abbas F. H. Alharan, István Kósa, Brigitta Szálka, István Vassányi

Abstract: Diet is a major, controllable risk factor for chronic disease management, therefore computerized meal plan generation is a key area of research. The paper presents a method for the automated mining of harmony rules that can be applied in the knowledge base of a meal plan generator. The proposed method is based on the extraction of co-occurrence patterns of food sets from dietary logs recorded by telemedical patients. The patterns are stored as knowledge graphs and assessed by several criteria (support, lift and coverage) to identify the most relevant ones. In a proof-of-concept study, the method was applied to generate a de facto rule set using a dietary log database of 20,373 meals, from which the best rule was used to extend the rule base of a meal generator framework, improving the quality of the generated meals.

Keywords: dietary log, knowledge graph, pattern discovery, rule mining, evolutionary computation

1 Introduction

Diet is a major contributor to chronic diseases. Early nutritional research concentrated on discrete elements, such as individual foods or nutrients [1], but subsequently moved toward comprehensive eating patterns that more accurately represent actual consumption [2]. As a result, modern dietary guidelines are concerned with overall consumption patterns [3], as health effects arise from interactions between foods and beverages [4]. These patterns vary across time and context, making meal planning complex [5]. Computerized meal plan generation is a useful tool for the tuning of personalized menus for dietitian experts and other users as well [6]. This process in most cases needs culture-specific expert dietary knowledge to be captured in a formalized knowledge base. Food compatibility is often represented by rules that provide positive or negative scores for food combinations. However, manually defining such rules is a tedious, difficult, and error-prone process, especially when attempting to capture logical patterns. Therefore, automated or interactive exploration of dietary patterns is essential [7]. Dietary data is often collected using food diaries, food frequency questionnaires (FFQs), and 24-hour dietary recalls, each of which has its own limitations: limited food coverage (FFQs), the need for data validation (food diaries), and a short-term scope (24-hour recalls) [8]. Despite pervasive use of data-driven approaches in dietary analysis, to the best of our knowledge, no prior study has explicitly modeled dietary patterns at the level of specific meal types. Most approaches of this type concentrate on overall eating habits, their relationships with health, and on prediction and personalization tasks. Therefore, this paper proposes a method for discovering meal-level dietary patterns in the form of interpretable compatibility harmony rules between food sets. The main hypothesis of this study was that the proposed rule mining process can find rules that can add value to a manually designed rule base.

2 Methods

As an input to the pattern mining, we used a dataset of 20,373 meals with 73,347 food records from a telemedical study. The food items appearing in the meals were organized in a hierarchy of 41 food sets by a dietitian expert. The main workflow of the proposed method consists of the following steps:

1. Data preprocessing and normalization, including cleaning missing or noisy entries, structuring dietary logs into meal-level records, and mapping food items to ontology-based food sets.

2. Knowledge graph (KG) construction with the main entities as patients, meals, foods and food sets.
3. Pattern mining on the knowledge graph using Cypher queries to identify frequent meal-specific food-set co-occurrence patterns and compute association measures in order to find the most relevant dietary patterns.
4. Translating the identified patterns into dietary rules for approval by a dietitian expert and adding the best rule to the rule base of a meal plan generator, i.e. the Multi-aspect Evolutionary Nutrition Designer (MEND) framework. The MEND rule base was assembled previously manually using a similar food and ontology base according to general dietary guidelines [9]. In order to check the usability of the new rule, meal plans generated with and without the rule were compared with respect to whether the meals in the generated plans conform more to the dietary pattern expressed by the rule.

As the main contribution of the paper, the pattern mining process is detailed below.

Set-level association mining

In order to identify the most important patterns of food sets, we use the concepts of support, confidence, and lift [10]. The *Support(A,B)* simply measures the relative frequency of the presence of two food sets A and B together in the same meal, across all meals. However, support may be high due to the mere popularity of the two sets without a relevant association, therefore we use the *Lift(A,B)* as the Support(A,B) divided by Support(A)*Support(B).

Lift favors associations that develop more than would be expected in a random chance setting. Furthermore, a possible source of bias in our dataset is that the dietary habits of patients with many recorded meals may be over-represented. Therefore, we define the *Coverage(A,B)* as a measure of how often patients consume the sets A and B together in a meal type:

$$\textit{Coverage}(A,B) = \frac{\#Patients(r_{A,B} \geq \tau)}{\#Patients(\text{MealType})}, \quad r_{A,B,t} = \frac{K_{A,B,t}}{T_t}$$

where $\#Patients(r_{A,B} \geq \tau)$ means the number of patients who "supported" the co-occurrence of sets A and B with at least a pre-defined threshold $\tau \in (0, 1]$ in the records of the relevant meal type, i.e. breakfast, lunch or dinner (left).

The "support" is defined by the co-consumption rate (right) computed for each patient, where the t is the meal type, T_t is the total number of meals of type t consumed by the patient, and $K_{A,B,t}$ is the number of those meals that contain both food sets A and B.

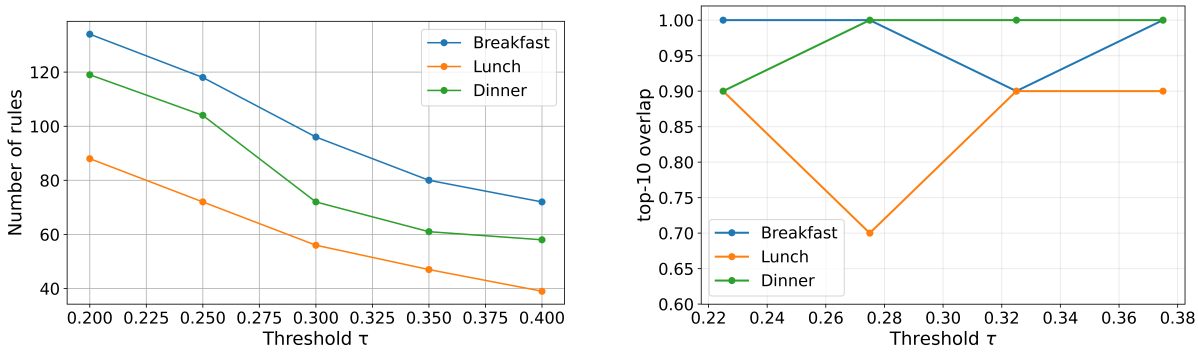
The goal of applying a thresholded co-consumption rate is to distinguish consistent (habitual) co-consumption from incidental co-consumption at the patient level.

Selection of the best rules

During the tests, we evaluated the sensitivity of the patient-normalized support threshold (τ) by repeating the mining process for each meal type using multiple threshold values and comparing the resulting rule sets in terms of rule count, Spearman rank correlation and Top-10 overlap ratio, i.e. as the ratio of the strongest 10 rules that are shared between two rule sets. The rule-selection process was based on the lift and patient-level coverage values. Lift was employed to retain only statistically dependent food-set pairs, whereas coverage is constrained within a predefined interval to eliminate both rare incidental combinations and overly generic associations dominated by highly frequent sets. The technologies used to implement the log database, the KG and the mining framework included Postgres, Neo4J and Python.

3 Results and discussion

Figure 1 (a) shows the number of mined rules across varying τ thresholds for the 3 meal types. As expected, rule counts decrease rapidly as τ increases, since stricter thresholds remove weak and rare co-occurrence patterns. The top-10 overlap ratio between consecutive τ levels (τ_i vs. τ_{i+1}) remains high across all comparisons (Figure 1 (b)), ranging from 0.7 to 1.0. This indicates that most strong rules are preserved as the threshold increases, confirming a strong stability of the mined rule structures. As a similar measure, Table 1 summarizes the Spearman rank correlation between rule-strength rankings obtained at consecutive τ levels for each meal type. Correlation values remain high across all meal types (> 0.94), indicating strong preservation of rule rankings as τ increases. The high Top-10 overlap and strong Spearman values (> 0.94) confirm that the main rules remain stable, thus the selected rules are statistically meaningful, representative at the patient level, and robust across threshold variations.



(a) Rule counts vs threshold values for Breakfast, Lunch, and Dinner meal types

(b) Overlap ratios of the strongest rule vs threshold values for Breakfast, Lunch, and Dinner meal types

Figure 1: Comparison of rule statistics across meal types

The final rules were selected according to the following criteria:

1. Lift ≥ 1 to ensure real statistical dependence.
2. Coverage limited to [0.10, 0.60] and $\tau \geq 0.1$ to remove rare or overly general patterns.

Table 1: Spearman Correlation between Consecutive τ Levels across Meal Types

$\tau 1$	$\tau 2$	Breakfast	Lunch	Dinner
0.20	0.25	0.966	0.944	0.947
0.25	0.30	0.940	0.837	0.949
0.30	0.35	0.984	0.865	0.971
0.35	0.40	0.974	0.937	1.00

The final result set contained 9 rules for breakfast, 3 for lunch and 5 for dinner. These rules were checked by a dietitian expert and those rules that contained sets known as risk factors for common diseases were dropped. These sets were the Meat products and Red meat sets. All the other rules were assessed by the expert as valuable and worth including in the knowledge base of an institutional meal plan generator. However, some other rules were also dropped because they used the Whole meal bread set which is not supported by the food database of MEND, or because according to the institutional constraints, foods in the Fresh vegetables set cannot appear in a breakfast or dinner. Thus, a lunch rule was selected for inclusion in the MEND rule base for the proof-of-concept test as the best rule. This rule favors the co-occurrence of Vegetable dishes and Poultry. Though a similar rule did already exist in the rule base, it referred to meat dishes in general, so the new rule can be regarded as a specialization.

In the last step, the new rule was added to the rule base of MEND, MEND was run several times, and the co-occurrence rate of the Vegetable dishes and Poultry sets in the generated plans was measured with, and without the activation of the new rule. The measured rates were 3.3% without, and 10.0% with the new rule.

4 Conclusion

We proposed and validated a KG-based method to extract reliable meal-specific dietary preference rules from patient eating logs. By applying patient-normalized mining, the approach identifies stable and meaningful food-set combinations that represent shared eating patterns across patients. The proposed approach was validated in a proof-of-concept study in a meal plan generator. The method could be easily generalized to mine more complex patterns including co-occurrences of more than two sets, temporal 'what-follows-what' patterns, or food frequencies over longer durations.

Acknowledgements

We acknowledge the Digital Government Development and Project Management Ltd. for awarding us access to the Komondor HPC facility based in Hungary.

References

[1] H. Ahsan. A brief overview and history of human nutrition and health. *Current Biochemistry*, 9(2), pp. 98–103, 2022.

[2] C.-A. Schulz, K. Oluwagbemigun, and U. Nöthlings. Advances in dietary pattern analysis in nutritional epidemiology. *European journal of nutrition* ,60(8), pp. 4115–4130, 2021.

[3] C. Agostoni, S. Boccia, G. Graffigna, J. Slavin, M. Abodi, and H. Szajewska. What should I eat today? evidence, guidelines, dietary patterns and consumer's behavior. *European journal of internal medicine*, vol. 126, pp. 26–32, 2024.

[4] Asfiya Meenaaz and Syeda Nishat Fathima. A review on food-food interactions. *Magna Scientia Advanced Research and Reviews*, 8 (1), pp. 123–127, 2023.

[5] M. F.-F. Chong. Dietary trajectories through the life course: opportunities and challenges. *British Journal of Nutrition*, 128(1), pp. 154–159, 2022.

[6] D. Tsolakidis, L. P. Gymnopoulos, and K. Dimitropoulos. Artificial intelligence and machine learning technologies for personalized nutrition: A review. *Informatics*, 11(3), 62, 2024.

[7] J. M. Hutchinson et al.Advances in methods for characterising dietary patterns: a scoping review. *British Journal of Nutrition*, vol. 133, no. 7, pp. 987–1001, Apr. 2025.

[8] R. L. Bailey. Overview of dietary assessment methods for measuring intakes of foods, beverages, and dietary supplements in research studies. *current opinion in biotechnology*, vol. 70, pp. 91–96, Aug. 2021.

[9] A. F. H. Alharan and I. Vassányi. Automated meal planning using ontologies and rules in evolutionary search. In *25th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 575–580, IEEE, Nov. 2025.

[10] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2), pp. 207–216, 1993.

Fine-Tuning CodeBERT-Java for Method-Level Bug Detection on Defects4J

Norbert Sándor Szolnoki, Gábor Antal

Abstract: We evaluate whether a pretrained transformer model can serve as an effective baseline for method-level bug detection in Java. This paper investigates the use of `CodeBERT-Java` for method-level bug detection on a dataset derived from `Defects4J`. We formulate the task as a binary classification problem in which Java methods are labeled as buggy or fixed, and fine-tune the pretrained `neulab/codebert-java` model for 15 epochs. The best validation balanced accuracy reaches 0.6053. On the test set of 392 samples, the model achieves 0.6301 accuracy, 0.6250 balanced accuracy, 0.6014 precision, 0.8259 recall, 0.6960 F1-score, 0.7007 ROC-AUC, and 0.2737 MCC. The results indicate that pretrained code language models can effectively capture defect-related information in Java methods, especially for identifying buggy instances, while also highlighting remaining challenges in class discrimination. These findings support the use of pretrained transformer models as a practical baseline for automated bug detection on `Defects4J`.

Keywords: Java, Defects4J, CodeBERT-Java, automated bug detection, method-level classification

1 Introduction

Software bugs remain a major source of failures, maintenance effort, and reduced reliability. As software systems grow in size and complexity, manually identifying buggy code becomes increasingly expensive, which motivates automated defect prediction and bug detection [1]. Recent pretrained code models have made it possible to learn code representations directly from large corpora, and models such as CodeBERT have shown strong results on a range of code understanding tasks [3].

In this paper, we study method-level bug detection with `CodeBERT-Java` on a dataset derived from `Defects4J`, a benchmark of real faults from Java projects [2]. We compare a zero-shot setting with a fine-tuned setting in order to determine whether task-specific supervision is necessary for practical defect detection.

The study addresses the following research questions:

- **RQ1:** How effective is fine-tuned `CodeBERT-Java` for method-level bug detection?
- **RQ2:** How well does the model balance buggy-method detection and overall classification quality?
- **RQ3:** How much does fine-tuning improve performance over the zero-shot baseline?

Our contribution is an empirical evaluation showing that fine-tuning is the key factor that turns a pre-trained Java code model into a usable bug detector.

2 Related Work

Automated defect prediction has traditionally relied on hand-crafted software metrics and classical machine learning methods, although their effectiveness often depends on the chosen features, projects, and evaluation setting [1]. For Java fault studies, `Defects4J` has become a standard benchmark because it provides real bugs and fixes from open-source projects in a controlled and reusable form [2]. It is now widely used in testing, automated repair, and bug-related learning studies [9, 10, 11].

More recently, pretrained transformer models have shifted the focus from manual feature design to learned code representations. CodeBERT showed that contextual embeddings can transfer effectively to downstream software engineering tasks [3], while CuBERT, GraphCodeBERT, and CodeT5 further improved code understanding through richer pretraining and structural information [4, 5, 7, 6]. Prior work has also shown that pretrained code models are promising for defect-related tasks, but usually require task-specific adaptation to perform well [8, 12, 13]. Our work builds on this direction by evaluating `CodeBERT-Java` for method-level bug detection on a Defects4J-derived dataset in both zero-shot and fine-tuned settings.

3 Methodology

We formulate method-level bug detection as a binary classification task over Java methods extracted from bug-fix revisions in `Defects4J` [2]. Starting from 16 projects, we compare buggy and fixed program versions, identify modified Java files, and then locate methods overlapping the changed lines. Each extracted method is treated as one instance and labeled as buggy ($y = 1$) if it comes from the buggy revision, or fixed ($y = 0$) if it comes from the corrected revision. After preprocessing and deduplication, the final dataset contains 2,574 labeled methods.

Given a method x, the model estimates the probability that it is buggy as

$$\hat{y} = P(y = 1 \mid x) = \sigma(h_\phi(f_\theta(x))) \,,$$

where f_θ is the pretrained `CodeBERT-Java` encoder, h_ϕ is a task-specific classification layer, and σ is the sigmoid function. This setup allows the model to learn directly from source-code sequences rather than from manually engineered software metrics.

We compare two settings: a zero-shot baseline without supervised adaptation and a fine-tuned setting trained on the same dataset. For fine-tuning, the data are randomly split into 70% training, 15% validation, and 15% test subsets using seed 42. Training is run for 15 epochs, and the best checkpoint is selected according to validation performance. The optimization objective is binary cross-entropy,

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right] .$$

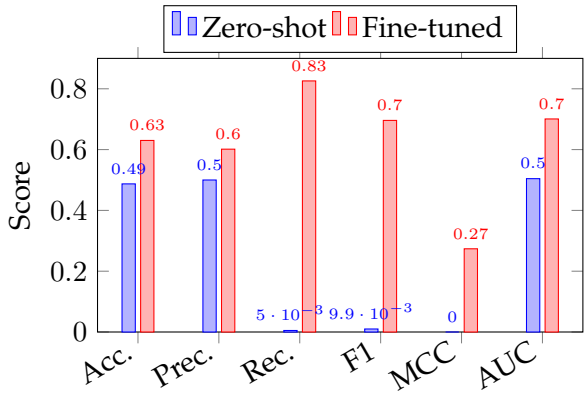
Final evaluation is performed on the held-out test set using accuracy, precision, recall, F1-score, ROC-AUC, and Matthews correlation coefficient (MCC)[15]. These metrics are used to assess not only overall classification quality, but also the trade-off between correctly detecting buggy methods and avoiding false alarms.

4 Results and Discussion

On the 392-sample test set, the zero-shot model achieves 0.4872 accuracy, 0.5000 precision, 0.0050 recall, 0.0099 F1-score, -0.0018 MCC, and 0.5044 ROC-AUC. After fine-tuning, performance rises to 0.6301 accuracy, 0.6014 precision, 0.8259 recall, 0.6960 F1-score, 0.2737 MCC, and 0.7008 ROC-AUC. Figure 4 summarizes the change.

RQ1. Fine-tuned `CodeBERT-Java` is a meaningful baseline for method-level bug detection, while the zero-shot model is not. The jump from 0.0099 to 0.6960 F1-score and from 0.5044 to 0.7008 ROC-AUC shows that supervised adaptation is essential.

RQ2. Fine-tuning changes the prediction behavior substantially. In the zero-shot setting, the model predicts almost everything as fixed, yielding only one true positive. After fine-tuning, recall rises to 0.8259 with 166 true positives, but this comes with 110 false positives. The model therefore favors sensitivity to buggy methods over conservative classification of fixed ones.



(a) Aggregate metrics

Setting	TP	FP	TN	FN
Zero-shot	1	1	190	200
Fine-tuned	166	110	81	35

(b) Confusion matrix summary

Figure 1: Comparison between zero-shot and fine-tuned `CodeBERT-Java`.

RQ3. The improvement over the zero-shot baseline is large across all aggregate metrics, indicating that the pretrained representation alone is insufficient for this task. Fine-tuning allows the model to learn defect-related signals that are not available in the zero-shot setting.

5 Threats to Validity

The study uses a `Defects4J`-derived dataset, so the results may not generalize to other languages, industrial systems, or defect categories. The binary buggy-versus-fixed formulation also simplifies real bug detection, where wider program context may matter. In addition, performance may depend on dataset construction choices, random splitting, and the use of a single pretrained model and training configuration. Different architectures, balancing strategies, or project-wise evaluation settings may produce different results.

6 Conclusion

This paper evaluated `CodeBERT-Java` for method-level bug detection on a `Defects4J`-derived dataset. The zero-shot model performed near chance, whereas fine-tuning substantially improved all main metrics, reaching 0.6960 F1-score and 0.7007 ROC-AUC. These findings show that pretrained code representations become useful for this task only after supervised adaptation. Fine-tuned `CodeBERT-Java` therefore provides a practical baseline for Java bug detection, while reducing false positives remains an important direction for future work.

Data Availability

The code package for constructing and evaluating the method-level Java bug detection dataset is publicly available on Zenodo [16].

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.

[2] R. Just, D. Jalali, and M. D. Ernst, "Defects4J: A Database of Existing Faults to Enable

Controlled Testing Studies for Java Programs," in *Proceedings of the International Symposium on Software Testing and Analysis*, pp. 437–440, 2014.

[3] Z. Feng *et al.*, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1536–1547, 2020.

[4] A. Kanade, P. Maniatis, G. Balakrishnan, and K. Shi, "CuBERT: Learning and Evaluating Contextual Embedding of Source Code," in *Proceedings of the ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[5] D. Guo *et al.*, "GraphCodeBERT: Pre-training Code Representations with Data Flow," in *International Conference on Learning Representations*, 2021.

[6] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified Pre-training for Program Understanding and Generation," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2655–2668, 2021.

[7] Y. Wang, W. Wang, S. Joty, and S. C. H. Hoi, "CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 8696–8708, 2021.

[8] M. Tufano, C. Watson, G. Bavota, M. Di Penta, M. White, and D. Poshyvanyk, "An Empirical Study on Learning Bug-Fixing Patches in the Wild via Neural Machine Translation," *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 4, pp. 1–29, 2019.

[9] H. Tian, K. Liu, A. K. Kaboré, A. Koyuncu, L. Li, and M. Monperrus, "Evaluating Representation Learning of Code Changes for Predicting Patch Correctness in Program Repair," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 981–992, 2020.

[10] E. Mashhadi and H. Hemmati, "Applying CodeBERT for Automated Program Repair of Java Simple Bugs," in *Proceedings of the 18th International Conference on Mining Software Repositories*, pp. 505–509, 2021.

[11] W. Zhong, C. Li, K. Liu, J. Ge, B. Luo, and M. Monperrus, "Benchmarking and Categorizing the Performance of Neural Program Repair Systems for Java," *ACM Transactions on Software Engineering and Methodology*, 2024.

[12] M. Abu Talib, A. Bou Nassif, M. Azzeh, Y. Alesh, *et al.*, "Parameter-Efficient Fine-Tuning of Pre-Trained Code Models for Just-in-Time Defect Prediction," *Neural Computing and Applications*, 2024.

[13] Y. Xiao, X. Zuo, L. Xue, K. Wang, J. S. Dong, and Y. Zhang, "Empirical Study on Transformer-Based Techniques for Software Engineering," *arXiv preprint arXiv:2310.00399*, 2023.

[14] Q. Zhang, C. Fang, Y. Xie, Y. Zhang, Y. Yang, *et al.*, "A Survey on Large Language Models for Software Engineering," *arXiv preprint arXiv:2312.15223*, 2023.

[15] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochim. Biophys. Acta*, vol. 405, no. 2, pp. 442–451, 1975.

[16] A. Anonymus and A. Anonymus, "Fine-Tuning CodeBERT-Java for Method-Level Bug Detection on Defects4J," 2026. [Online]. Available: `https://doi.org/10.5281/zenodo.19087509`

End-to-End Verified Parking Trajectory Prediction with Neural Networks

Nándor Ákos Vincze, Balázs Bánhelyi

Abstract: Reliable trajectory generation is essential for autonomous parking systems, particularly as vehicle automation advances towards Level 4 autonomy, which requires systems to assume full responsibility without human intervention. This work presents an end-to-end verified pipeline is presented that combines numerical optimization, compact neural networks, and formal numerical verification to predict steering parameters. First, optimal parking trajectories are generated by solving a vehicle motion model using interval arithmetic to mitigate discretization and floating-point errors. A parametric steering function is optimized to produce numerically stable trajectories. The resulting dataset is used to train a multi-layer perceptron (MLP) consisting of 32 neurons. To ensure safety and overcome the "black-box" nature of neural networks, the network's predictions are verified using interval and affine arithmetic, allowing the evaluation of trajectories under bounded input uncertainty. The verification procedure identifies regions of the state space where the predicted maneuvers are provably collision-free. The proposed end-to-end verified system is demonstrated for reverse parallel parking maneuvers. and the results demonstrate that compact and formally verifiable neural networks can provide fast and reliable trajectory prediction for autonomous parking applications.

Keywords: Reliable trajectory, autonomous parking systems, Neural Network, Verification

1 Introduction

As autonomous vehicle technology advances, reliable parking maneuver planning remains a critical challenge due to diverse and crowded environments [3]. Deep convolutional networks are difficult to formally verify due to their black-box nature. This work proposes a verifiable, 32-neuron fully connected model. By modeling vehicle motion with differential equations and employing interval computations, we eliminate floating-point arithmetic errors. The model is then formally verified using interval arithmetic to guarantee robustness against input variations.

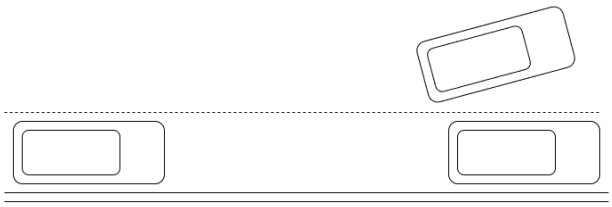


Figure 1: The modeled parking situation.

2 Model and Methods

2.1 Parking Model

The considered maneuver is a reverse parallel parking task, where the vehicle moves backwards into a space located between two stationary vehicles. Assuming a constant speed of $1\ \mathrm{m/s}$, the trajectory depends entirely on the steering angle. The steering function at time t is defined as: $\phi(t) = \frac{1}{3}\arctan(at + b)$

The trajectory is described using a kinematic bicycle model [2]. Let $X : [0, T] \to \mathbb{R}^3, \quad X(t) = (x(t), y(t), \theta(t))^T$ be the state function describing the vehicle's position. The system is formulated as an initial value problem (IVP):

$$F(t, X) = \begin{pmatrix} \cos(\Theta) \\ \sin(\Theta) \\ -\frac{1}{w} \tan\left(\frac{1}{3} \arctan(at + b)\right) \end{pmatrix}$$

$$\begin{cases} X' = F(t, X) \\ X(0) = (x_0, y_0, \theta_0)^T \end{cases}$$

2.2 Interval Arithmetic

Standard differential equation solvers utilizing floating-point arithmetic (e.g., Runge-Kutta) evaluate the system at discrete points in time which can cause the simulation to "miss" boundary crossings between discrete steps. To handle this, interval arithmetic is applied. By enclosing state variables in intervals, the simulation guarantees the capture of intermediate states and rounding errors. The Taylor model (via the INTLAB package [4]) is utilized to mitigate the dependency problem and wrapping effect.

2.3 Optimization

The goal is to determine the optimal parameters (a, b) and maneuver time $T > 0$ for a given starting position. A comprehensive objective function $J : \mathbb{I}^3 \to \mathbb{I}$ is constructed to penalize boundary violations and collisions. The base cost function aggregates a heavy collision penalty ($P_{collision}$), a parking space boundary penalty ($P_{parking}$), a distance metric (d_{metric}), and terminal state errors (e_x, e_y, e_θ):

$$J_{base} = 10^4 P_{collision} + 10^3 P_{parking} + 0.5 d_{metric} + 25 e_x^2 + 25 e_y^2 + 50 e_\theta^2$$

The final objective function also accounts for uncertainty by adding the diameter of the evaluated interval as well as the uniqueness of the steering function:

$$J = J_{base} + \text{diam}(J_{base}) + a^2 + b^2$$

The derivative-free Nelder-Mead simplex algorithm is applied to minimize this objective function.

2.4 Training Dataset

To train the neural network, a dataset of approximately 10,000 samples was generated across a predefined spatial domain: $x \in [48.5, 54]$, $y \in [-10.62, -8]$, and $\theta \in [-0.8, 0.2]$. The PyTorch framework [1] was used to train a small, 32-neuron network to predict (a, b, T) based on the initial state. The small network architecture prevents overfitting and ensures that interval widths remain manageable during verification.

3 Verification of the Neural Network

Verification proves that no prediction leads to a collision or unsafe trajectory within the neighborhood of an input triplet (x, y, θ). Because vehicle sensors operate with finite accuracy and environmental noise, uncertainty must be modeled at the input.

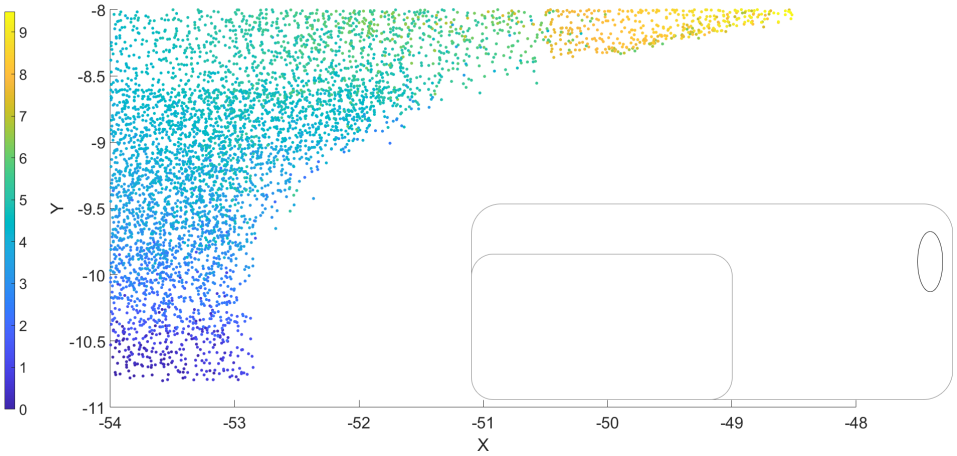


Figure 2: The parking time displayed from different starting coordinates (X,Y) in the dataset, with one of the stationary cars displayed in gray.

During verification, lower and upper bounds are specified to yield an interval vector $(A, B, \hat{T}) \in \mathbb{I}^3$. Because naive interval arithmetic causes severe overestimation, affine arithmetic is utilized via the `estimateNetworkOutputBounds` function in the MATLAB Deep Learning Toolbox, which tracks dependency sources and provides tighter bounds.

A grid search algorithm evaluates the parameters in small spatial regions of 10 cm $\times$ 10 cm $\times$ 0.05 rad. If the evaluated objective function satisfies $\sup(J(A, B, \hat{T})) < 100$, the spatial region is considered verified. By fitting these regions together, a continuous, reliable operating domain is established.

4 Results

The developed pipeline successfully combines optimization, neural network training, and formal verification (Figure 1). The verification process identified secure spatial cells within the training domain where collision-free maneuvers are strictly guaranteed.

A significant advantage of this approach is real-time performance. While classical optimization with a reliable differential equation solver takes approximately 34.2 seconds to calculate a trajectory, the floating-point evaluation of the trained neural network completes in just 0.0158 seconds.

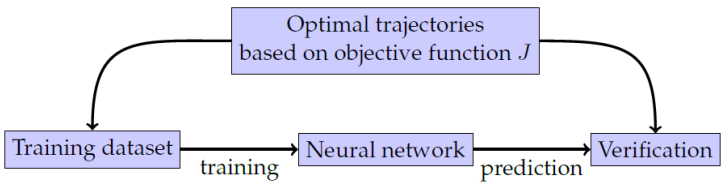


Figure 3: Overview of the complete process

The part of the area where the network's predictions guarantee a collision-free maneuver is illustrated in Figure 4, which shows the projection onto the $x - y$ plane. These regions confirm that if a measurement error occurs but the error remains within the verified domain, the network will continue to operate safely.

Because the verified region is available near the parking space, **iterated evaluation** becomes possible. The optimal steering parameters can be re-queried (e.g., every h seconds) using the

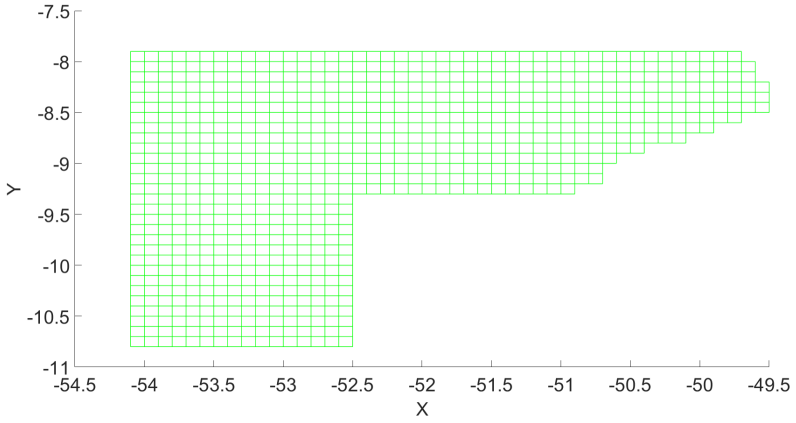


Figure 4: Projection of the verified spatial domain onto the $x - y$ plane.

vehicle's momentary position. This dynamic correction process increases the robustness of the maneuver, compensating for any physical deviations (Figure 5).

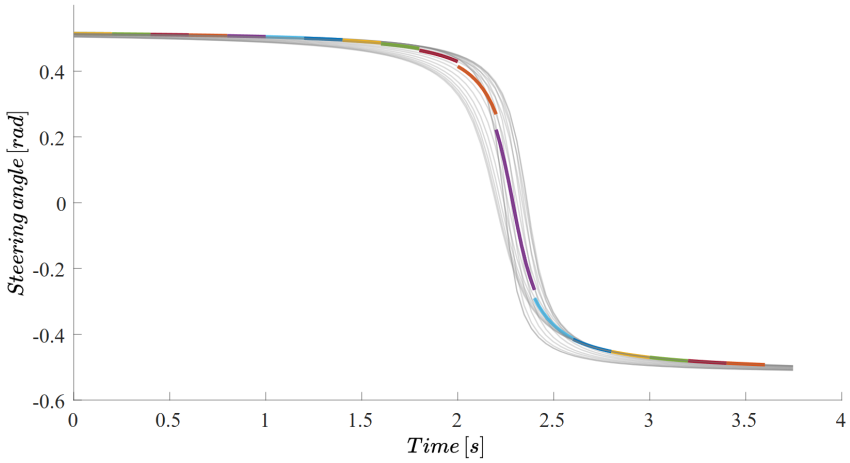


Figure 5: Iterated evaluation of the steering parameters during a single maneuver.

References

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. arXiv preprint arXiv:1912.01703, 2019.

[2] P. Polack, F. Altché, B. Novel and A. de La Fortelle, *The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?* In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 812–818. doi: 10.1109/IVS.2017.7995816

[3] SAE International, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (J3016)*. SAE International, 2021. DOI: 10.4271/J3016_202104.

[4] S. M. Rump, *INTLAB - INTerval LABoratory*. In: T. Csendes (Edited by), Developments in Reliable Computing. Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104.

AI Agent-based Approach for Enhancing Urban Mobility Applications

Zahra Delavari

Abstract: Rapid urbanization has intensified the need for efficient, sustainable, and personalized mobility solutions, while existing travel planning systems remain largely static and limited in adaptability. This paper explores the role of AI agents, particularly those powered by large language models (LLMs), in addressing these challenges. It reviews current AI agent architectures and frameworks, highlighting their capabilities in reasoning, tool integration, and multi-agent collaboration. Building on these insights, the paper proposes a conceptual architecture for an intelligent urban travel agent that integrates strategy management, structured reasoning, tool orchestration, memory systems, and real-time adaptation. The design emphasizes personalization, explainability, and continuous learning to improve user experience and support environmentally conscious decision-making. Overall, the work demonstrates the potential of AI agents to transform urban mobility planning into a more adaptive, user-centric, and sustainable process.

Keywords: Urban Travel Planning, Multi-agent orchestration,Large Language Models (LLMs) and Planning, Personalized Travel Planning, Retrieval-Augmented Generation (RAG)

1 Introduction

Urbanization has rapidly increased the demand for efficient transportation in large cities. The rise in travel for work, education and services has made mobility planning essential, but challenges such as traffic congestion, long commutes, delays, pollution, and reduced productivity persist, exacerbated by widespread car use [2]. Private vehicles contribute significantly to greenhouse gas emissions, harming air quality and climate, highlighting the need for environmentally friendly travel options [3]. Traditional route planners rely on fixed criteria such as travel time or distance [4], often lacking adaptive reasoning, personalization, and consideration of environmental impact, user preferences, or real-time disruptions [5].

Recent advances in AI, especially large language models (LLMs), have enabled intelligent agents to integrate data sources, services, and reasoning [7]. AI agents are software systems that perceive their environment, make decisions and act to achieve goals [10]. Compared to traditional AI, they are more autonomous and adaptive, learning from interactions and adjusting strategies to handle complex, dynamic tasks. This paper presents a conceptual architecture for an AI agent that supports sustainable, personalized urban travel and compares existing AI agent frameworks for such applications.

Agent architectures can be single-agent or multi-agent. Single-agent systems typically include planning, reasoning, memory, action execution, and environment interaction, supporting goal-oriented behavior and continuity through memory mechanisms [11]. Multi-agent systems consist of specialized agents that collaborate on complex tasks. Agents coordinate through real-time communication and feedback, improving flexibility, robustness, and scalability. Architectures can be hierarchical, with a central agent managing others, or collaborative, where agents operate equally. Common designs include centralized, distributed, and hybrid models, each with trade-offs in control and scalability [12]. Several frameworks have been developed to simplify AI agent development, providing features such as tool integration, memory management, and multi-agent coordination Table 1.

Table 1: AI Agent Frameworks

Framework	Key Features	Applications	Advantages	Limitations
LangChain	LLM orchestration, tool calling, memory modules, retrieval-augmented generation (RAG), multi-agent workflows	Building LLM-powered applications; multi-agent LLM apps; conversational agents	Highly modular; supports stateful workflows; flexible tool integration	Complex for beginners; requires integration effort
AutoGen	Multi-agent conversation framework, autonomous agents, tool integration, collaborative task solving, code execution	Multi-agent systems completing complex tasks; autonomous assistants; complex workflows	Strong multi-agent reasoning; supports collaborative task solving	Requires Microsoft ecosystem; higher setup complexity
CrewAI	Role-based agents, task delegation, workflow coordination, multi-agent collaboration	Structured team-based agents performing coordinated tasks	Efficient collaboration; structured roles	Less flexible for single-agent tasks
MetaGPT	Role-based agent architecture, multi-agent collaboration, structured task pipelines	AI-driven software development; complex project management	Structured pipelines; supports multi-agent collaboration	Niche focus; less general-purpose flexibility
Semantic Kernel	AI orchestration framework, plugin architecture, memory integration, tool execution; enterprise SDK	Enterprise AI applications; integrating LLMs with existing services	Enterprise-ready; plugin support; workflow integration	Dependent on Microsoft ecosystem; steep learning curve
Langflow	Visual interface for building LLM pipelines; tool integration; workflow management	Rapid prototyping of AI agents and LLM workflows	Visual interface simplifies prototyping; easy workflow management	Less suited for complex multi-agent orchestration
n8n	Workflow automation platform; API integration; event-based triggers	Automating workflows; connecting AI services with other software systems	Strong automation capabilities; integrates with many services	Not specialized for LLMs; limited AI reasoning
LlamaIndex	Data-centric agents; RAG	Data querying; document reasoning	Excellent for data-intensive tasks	Limited multi-agent orchestration
Pydantic AI	Python-centric; type-safe; durable execution	Production-ready agents; API integration	Reliable; production-ready	Less flexible for rapid prototyping
Smolagents	Minimalist agent loop	Quick automation; prototyping	Lightweight; fast prototyping	Limited features; basic orchestration
OpenAI Swarm	Educational/experimental; multi-agent orchestration	Research, teaching	Lightweight; easy experimentation	Not production-ready; limited documentation
AgentLite	Lightweight research library; manager-agent orchestration	Research prototyping; urban travel simulations	Simple; flexible for experimentation	Not production-ready; small ecosystem

2 Related Agent-based Solutions for Travel Planning

Despite rapid advances in AI agents, the travel and hospitality sectors continue to lag in adoption. Most existing solutions focus on general-purpose copilots and chatbots with limited impact, while domain-specific applications remain largely in early stages [1].

Trip planning is traditionally a time-consuming process that requires collecting information from multiple sources. Many existing tools provide static and non-personalized content, whereas recent approaches leverage recommendation engines and automation to offer more tailored and efficient planning [14]. Although research on AI agents specifically for travel is still limited, several recent works demonstrate their potential. AgentTravel [15] uses a loop of reasoning-action with integration of tools to generate travel plans. Eco-friendly route guidance systems [16] apply multi-agent architectures to address traffic and environmental challenges. TraveLLaMA [17] improves multimodal understanding by combining visual and textual travel data. Travel Optix [14] employs multiple collaborative agents to automate itinerary generation, while TravelAgent [18] integrates reasoning, memory, and planning capabilities for more comprehensive decision-making. Similarly, VAIAGE [19] utilizes a graph-based multi-agent system with real-time data integration. In general, these approaches demonstrate a shift towards more intelligent and collaborative systems; however, they remain limited in providing a unified, scalable architecture for urban travel planning.

[1] https://www.mckinsey.com/

3 The proposed AI Agent-based Mobility Architecture

Our proposed AI agent-based approach consists of several interconnected components that enable intelligent and adaptive urban travel planning:

- **Manager / Strategy Agent:** Serves as the structural foundation of the system. Interprets user goals, coordinates the different components, and ensures consistent decision-making. This agent receives user requests, calls external routing and data tools, makes final decisions, and manages other agents if present, ensuring smooth collaboration and orchestration [12][13].
- **Reasoning Strategy:** Enables the agent to analyze multiple route options systematically rather than producing arbitrary recommendations. By structuring decision-making, it prevents random choices, makes agent behavior explainable, and supports debugging and performance evaluation [17].
- **Tool Orchestration:** Manages interaction with external services such as routing engines, traffic feeds, and environmental data sources. Instead of computing routes internally, the agent leverages structured outputs from tools and APIs. This approach keeps the system modular, scalable, and easier to maintain [13].
- **Memory System:** This allows the agent to retain contextual information, improving personalization and recommendation quality over time. It includes short-term memory—covering the current trip, the current route, and recent events—and long-term memory, which stores user preferences, past route choices, and accepted or rejected suggestions. This dual-memory design reduces unnecessary rerouting, supports personalized decisions, and allows the agent to learn user trade-offs over time [10][14].
- **Real-Time Adaptation:** Ensures the agent responds effectively to disruptions such as congestion or delays. employing event-driven reasoning, the system can dynamically adjust routes and strategies, providing timely alternatives that maintain user satisfaction and efficiency [14][16].
- **Explanation & Avatar:** Maintains transparency and trust by communicating the agent's reasoning to the user. Through a conversational user-facing interface, the agent explains its decisions, justifies trade-offs, requests consent, and answers *why* questions, ensuring clarity and usability [18].
- **Evaluation & Learning:** Monitors system performance to support continuous improvement. Metrics include reroute acceptance rates, user rejection frequency, and overall trust or satisfaction. Combined with detailed logs of decisions and outcomes, this component allows the agent to refine its strategies and improve recommendations over time [19].

4 Conclusions

In this paper, we examined the potential use of AI agents, particularly those powered by large language models, to address key challenges in urban mobility planning. We reviewed existing agent methods and frameworks, and identified their strengths and limitations in supporting personalized, adaptive, and sustainable travel solutions. Based on this analysis, we proposed a conceptual AI agent-based architecture that integrates reasoning, tool orchestration, memory, and real-time adaptation to enhance decision-making and user experience. The proposed approach represents our first step in our planned research, and highlights the importance of explainability, scalability, and continuous learning in next-generation travel systems. As future

work, we aim to develop a portal for the SO-SMART EIG CONCERT-Japan project, building on the identified AI agent categories to create a practical, user-centered platform for intelligent urban mobility services.

References

[1] F. Canitez, P. Alpkokin, and S. Topuz Kiremitci, "Sustainable urban mobility in Istanbul: Challenges and prospects," *Case Studies on Transport Policy*, vol. 8, no. 4, pp. 1148–1157, 2020.

[2] V. K. Giduthuri, "Sustainable urban mobility: Challenges, initiatives and planning," *Current Urban Studies*, vol. 3, no. 3, pp. 261–265, 2015.

[3] L. Pan, N. Yang, L. Zhang, R. Zhang, B. Xie, and H. Yan, "Assessment of the Impact of Multi-Agent Model-Based Traffic Optimization Interventions on Urban Travel Behavior," *Electronics*, vol. 14, no. 1, 2025.

[4] T. Zhang and F. Gao, "A Traditional Cultural Route Planning and Design System Based on Improved Genetic Algorithm," *Procedia Computer Science*, vol. 247, pp. 211–217, 2024.

[5] S. Karhu, "Automating Multi-City Concert Travel Planning with AI Agents," 2025.

[6] T. Neumayr and M. Augstein, "A systematic review of personalized collaborative systems," *Frontiers in Computer Science*, vol. 2, p. 562679, 2020.

[7] S. Kamatala, "Ai agents and llms revolutionizing the future of intelligent systems," *Yes it was accepted in IJSRED published in*, vol. 7, no. 6, 2024.

[8] X. Huang, J. Lian, Y. Lei, J. Yao, D. Lian, and X. Xie, "Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations," *ACM Trans. Inf. Syst.*, vol. 43, no. 4, 2025.

[9] F. Zhang and B. Wu, "Large Language Models as General Purpose Intelligence Systems for Reasoning, Planning and Decision Making," *American Journal of Artificial Intelligence and Neural Networks*, vol. 6, no. 4, pp. 45–72, 2025.

[10] P. Zhao, Z. Jin, and N. Cheng, "An in-depth survey of large language model-based artificial intelligence agents," *arXiv preprint arXiv:2309.14365*, 2023.

[11] Y. Cheng *et al.*, "Exploring large language model based intelligent agents: Definitions, methods, and prospects," *arXiv preprint arXiv:2401.03428*, 2024.

[12] X. Zhang, X. Dong, Y. Wang, D. Zhang, and F. Cao, "A Survey of Multi-AI Agent Collaboration: Theories, Technologies and Applications," 2025.

[13] T. Masterman, S. Besen, M. Sawtell, and A. Chao, "The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey," *arXiv preprint arXiv:2404.11584*, 2024.

[14] A. Singh, R. Madhogaria, A. Misra, and E. Elakiya, "Automated travel planning via multi-agent systems and real-time intelligence," in *Proceedings of the 3rd International Conference on Optimization Techniques in the Field of Engineering (ICOFE-2024)*, 2024.

[15] J. Zhao, J. Feng, and Y. Li, "AgentTravel: Knowledge-Augmented LLM Agent Framework for Urban Travel Planning," 2025.

[16] A. Namoun, A. Tufail, N. Mehandjiev, A. Alrehaili, J. Akhlaghinia, and E. Peytchev, "An eco-friendly multimodal route guidance system for urban areas using multi-agent technology," *Applied Sciences*, vol. 11, no. 5, p. 2057, 2021.

[17] M. Chu, Y. Chen, H. Gui, S. Yu, Y. Wang, and J. Jia, "TraveLLaMA: A Multimodal Travel Assistant with Large-Scale Dataset and Structured Reasoning," *arXiv preprint arXiv:2504.16505*, 2025.

[18] A. Chen, X. Ge, Z. Fu, Y. Xiao, and J. Chen, "Travelagent: An ai assistant for personalized travel planning," *arXiv preprint arXiv:2409.08069*, 2024.

[19] B. Liu, J. Ge, and J. Wang, "Vaiage: A Multi-Agent Solution to Personalized Travel Planning," *arXiv preprint arXiv:2505.10922*, 2025.

Deterministic Diagnostics for Highly Configurable 5G Networks Using Inductive Learning

Balázs Toldi, András Földvári, Zsigmond Pap, András Pataricza

Abstract: 5G Core networks are constructed on a family-based principle to fit to the demands of the individual users and dimensioning of its components happens only at deployment time across all configuration combinations prior to the final acceptance test. This necessitates an extrapolation of the characteristics of the designated configuration based on particular benchmarking results from a few representative components. We propose a deterministic, predictive approach leveraging qualitative models extracted from such real-world measurements and simulation data. By applying qualitative abstraction to system telemetry, we enhance interpretability while drastically reducing model complexity; at the same time the simplification of the modelling space to qualitative values essentially preserves the main phenomena and results in a faithful prediction for an individual target configuration. Our method employs inductive learning to derive deterministic, qualitative rules that characterize system health and bottlenecks. Unlike probabilistic machine learning models, this approach yields human-readable, well-interpretable, logic guarantees suitable for critical infrastructure. The resulting framework provides a systematic method for evaluating model faithfulness, identifying configuration flaws, and ultimately reducing the required number of test cases through targeted rule verification.

Keywords: Inductive Learning, Explainable AI, Critical Infrastructure, Answer Set Programming

1 Introduction

Modern 5G Core networks are highly configurable, family-based systems that form part of critical infrastructure [5, 6]. They are deployed to fulfil vastly different user demands and must dynamically support diverse traffic profiles - from high-bandwidth applications to massive, low-power IoT device swarms. Because these networks form our critical infrastructure, their operation underlies strict service level requirements. However, ensuring this dependability is incredibly challenging. The variability in deployment architectures (e.g., scaling processing nodes, varying CPU computational power) and traffic demands creates a combinatorial explosion of possible system states [7]. Consequently, exhaustive testing of all configuration types is economically and computationally infeasible.

While traditional machine learning (ML) approaches are frequently successful for anomaly detection in such networks, they often fall short for critical systems. Standard ML models are inherently probabilistic and function as "black boxes," failing to provide operators with the definitive, interpretable explanations [8, 9] required to diagnose and resolve architectural flaws safely. To address this, we propose a deterministic, logic-based learning approach. By leveraging Inductive Learning, we autonomously generate human-readable, deterministic diagnostic rules from raw system telemetry data, providing the interpretability and dependability that validation of critical systems demand.

2 Methodology: Data Abstraction and Inductive Learning

A further complexity of modelling originates in the additional variability of expected workloads in addition to the functional and architectural composition and dimensioning of the target system. As a pilot to capture the varied states of a complex network without succumbing to combinatorial explosion, we constructed an emulated 5G core subjected to a matrix of operational variables. We systematically varied deployment capabilities (UPF counts and CPU

capacities) alongside diverse traffic demands (Massive IoT, eMBB, and standard traffic) and throughput targets.

Rather than feeding raw, high-dimensional time-series data directly into a statistical model over continuous domains, our approach relies on symbolic representation by abstracting the values into qualitative ordinals. This facilitates the evaluation of the quantized data by the huge repertoire offered by logic and discrete formal methods. This methodology consists of three distinct phases:

1. **Data Collection & Quantization:** Time-series metrics (CPU, memory utilization, throughput, and packet statistics) are collected, normalized, and quantized into discrete abstractions (e.g., LOW, MEDIUM, HIGH) [10, 3]. Critical error states, such as packet drop rates, are strictly categorized to establish clear, deterministic boundaries between healthy and unhealthy system states.

2. **Knowledge Base Construction:** The quantized data is transformed into a symbolic format. We aggregate the environmental states into positive and negative examples with respect to the requirement related to the expected service level to serve as the training foundation.

3. **Rule Synthesis via Inductive Learning:** To generate an exact phenomenological model of the system over the qualitative discrete domain, we utilize FASTLAS [4, 1] over Answer Set Programming (ASP; A prolog-style declarative reasoning paradigm supporting non-determinism and negation as failure) to process these examples via Inductive Learning. This model characterizes in a qualitative form, the resulting service quality as a function of the workload and dimensioning parameters. One advantage of this approach is that it can take into consideration the architecture and qualitatively measured system as well.

4. **Evaluation of the Results by a Domain Expert:** As the result of the evaluation is a compact logic function over the workload and the dimensioning parameters in a qualitative form, the result fits the common sense engineering thinking well to give very well interpretable and Explainable form of machine learning. For instance, the engineer faces expressions like packet drop rate is HIGH if the memory capacity is LOW or the memory capacity is MEDIUM, but the computing power is LOW. An additional advantage is that individual qualitative categories are already independent of the actual scale and dimensioning of the resources of the workloads and facilitates a kind of transfer learning from benchmarking results over a set of representative configuration to other ones.

3 Preliminary Results and Discussion

Initial experiments demonstrate that our logic-based approach can successfully identify complex environmental issues and bottlenecks purely from the data after abstraction, without requiring prior domain-specific guidance.

The Inductive Learning model generated a concise set of ASP rules that accurately and deterministically characterize the conditions leading to system failure. For example, the system autonomously identified architectural vulnerabilities, outputting rules such as:

- **Example rule 1:** A node is considered UNHEALTHY if its traffic type is categorized as mIoT (massive Internet of Things, characterized by small packets in the network with high frequency).

- **Example rule 2:** A node is considered UNHEALTHY if its CPU strength is rated as LOW.

These rules immediately informed domain experts that the current deployment configuration could not reliably handle Massive IoT (mIoT) traffic profiles, and that specific CPU constraints

were direct catalysts for failure. Because the output is inherently deterministic and human-readable, domain experts were able to instantly verify [2] the system's findings and pinpoint the exact misconfigurations causing the anomalies.

4 Model verification

Moving forward, this framework will serve as the foundation for automated model verification through cross-deployment testing. By deploying the inductively learned rule set against unseen network configurations, we can test the faithfulness and generalizability of the generated logic. Instances where the existing rules fail to predict a bottleneck will be captured and fed back into the system as new negative examples. This iterative refinement will allow the deterministic model to continuously evolve, informing case prioritization and drastically reducing the number of necessary test cases for future deployments.

5 Summary

In this paper, we addressed the challenge of diagnosing and validating highly configurable critical infrastructure, specifically 5G Core networks. Driven by the prohibitive computational cost of exhaustive combinatorial testing and the inherent opacity of probabilistic machine learning models, we introduced a deterministic, logic-based diagnostic framework. By abstracting multidimensional, time-series telemetry into qualitative states and applying inductive learning, we successfully synthesize human-readable diagnostic rules. Our preliminary results demonstrate that this approach can autonomously and accurately identify system bottlenecks—such as hardware constraints and traffic-specific vulnerabilities—without requiring prior domain knowledge. Ultimately, this explainable framework establishes a robust foundation for iterative model verification. It provides the definitive reliability essential for critical telecommunication systems while significantly reducing the required testing space for future deployments.

Acknowledgements

Project no. 2025-2.1.2-EKÖP-KDP-2025-00005 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the EKÖP_KDP-25-1-BME-27 funding scheme.

References

[1] M. Law, A. Russo, and K. Broda, "The ILASP System for Inductive Learning of Answer Set Programs."

[2] F. Wotawa, "On the use of answer set programming for model-based diagnosis," in *Int. Conf. on Industrial, Eng. and Other Appl. of Applied Intelligent Sys.*, pp. 518–529, 2020.

[3] I. Kocsis, "Qualitative models in resilience assurance," 2019.

[4] M. Law, A. Russo, E. Bertino, K. Broda, and J. Lobo, "FastLAS: Scalable Inductive Logic Programming Incorporating Domain-Specific Optimisation Criteria," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2877–2885, 2020.

[5] M. Settembre, "A 5G Core Network Challenge: Combining Flexibility and Security," in *2021 AEIT International Annual Conference (AEIT)*, pp. 1–6, 2021.

[6] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, *5G Core Networks: Powering Digitalization*. Academic Press, 2019.

[7] J. Meinicke, C.-P. Wong, C. Kästner, T. Thüm, and G. Saake, "On essential configuration complexity: measuring interactions in highly-configurable systems," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pp. 483–494, ACM, 2016.

[8] D. S. Watson, "Conceptual challenges for interpretable machine learning," *Synthese*, vol. 200, no. 2, p. 65, 2022.

[9] M. Mukelabai, D. Nešić, S. Maro, T. Berger, and J.-P. Steghöfer, "Tackling combinatorial explosion: a study of industrial needs and practices for analyzing highly configurable systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 155–166, Association for Computing Machinery, 2018.

[10] K. D. Forbus, "Chapter 9 Qualitative Modeling," in *Foundations of Artificial Intelligence* (F. van Harmelen, V. Lifschitz, and B. Porter, eds.), vol. 3, pp. 361–393, Elsevier, 2008.

Comparing Large Language Models for Automated Vulnerability Repair under a Fixed Prompting Strategy

Vivien Vörös, Gábor Antal, Péter Hegedűs

Abstract: Recent advances in Large Language Models (LLMs) show promise in fixing vulnerable code, and as these models are frequently updated, their performance on real-world vulnerabilities remains not fully understood. In this paper, we evaluate the ability of GPT-4, GPT-4o, and GPT-5 to repair Java vulnerabilities using a subset of the VUL4J dataset. All models are tested with the same prompt setup to ensure a fair comparison. Since model outputs can vary, each model is run three times per sample. We measure performance using fix ratio, minimum and maximum number of fixes, and an at-least-once success rate. The results show that GPT-4 and GPT-4o achieve higher average repair performance, while GPT-5 behaves more consistently across runs. An analysis based on Common Weakness Enumeration (CWE) categories shows that no single model performs best across all vulnerability types. These results suggest that newer models do not always perform better when using the same prompts. They also highlight the importance of considering both effectiveness and consistency when applying LLMs to software security tasks.

Keywords: LLM, vulnerability repair, automated program repair, prompt engineering, software security

1 Introduction

Software vulnerabilities are a major challenge in modern software systems, potentially leading to security risks, data breaches, and system failures. Identifying and repairing these vulnerabilities manually is time-consuming and increasingly difficult as software systems grow in complexity. Recently, Large Language Models (LLMs) have shown promising potential in automating vulnerability detection and repair. [2, 1]

Automated vulnerability repair using LLMs has become an important research area. These models are capable of understanding source code and generating fixes for various types of issues, including security-related ones. However, not all LLMs perform equally well, and their effectiveness in handling real-world vulnerabilities can vary significantly.

To better characterize vulnerabilities, it is useful to consider CWE (Common Weakness Enumeration) [3] and CVE (Common Vulnerabilities and Exposures) [4]. CWE describes general categories of weaknesses, helping to understand common patterns of insecure coding, while CVE refers to specific, real-world vulnerability instances. The CVE collection, maintained by the NVD [6], provides widely referenced and frequently updated information, including descriptions, severity scores, and affected products, enabling reliable assessment and comparison of security flaws over time. Incorporating such structured and up-to-date information into prompts can guide LLMs to generate more accurate, context-aware, and robust fixes, thereby improving automated vulnerability repair in practice.

In this paper, we investigate the repair of Java vulnerabilities using the VUL4J [5] dataset, which contains real-world vulnerabilities along with human-written fixes. Each vulnerability includes a test suite with failing cases that are used to verify whether a generated patch correctly fixes the issue.

- **RQ:** How do different LLMs (GPT-4, GPT-4o, and GPT-5) compare in their ability to repair real-world Java vulnerabilities under a fixed prompting strategy?

We compare GPT-4, GPT-4o, and the base GPT-5 model (excluding GPT-5.1, GPT-5.3, and GPT-5 Mini) under a fixed prompt configuration for fair comparison. While GPT-4 and GPT-4o results are based on previous setups [1, 2], we extend the evaluation to GPT-5.

The goal of this work is to assess how effectively LLMs can repair vulnerabilities compared to human-written solutions, and to identify trade-offs between models when applied to real-world security issues.

2 Methodology

We evaluate the capability of Large Language Models (LLMs) to automatically fix security vulnerabilities in source code, building on the prompt and reported results from a preliminary studies: GPT-4 [1] and GPT-4o [2]. Specifically, we compare three models: GPT-4, GPT-4o, and GPT-5 under identical experimental conditions to assess both their effectiveness and consistency. By extending prior findings, we aim to provide a more comprehensive understanding of how model architecture and updates influence automated vulnerability repair performance.

2.1 Dataset

We selected 42 vulnerabilities from the VUL4J dataset [5], a curated collection of real-world Java vulnerabilities designed for research on program and vulnerability repair. In VUL4J, each vulnerability is paired with a human-written patch and supporting artifacts such as Proof-of-Vulnerability (PoV) test cases that fail on the vulnerable version and pass on the fixed version, enabling reproducible evaluation of repair techniques. The original dataset contains a set of reproducible vulnerabilities drawn from open-source projects spanning multiple Common Weakness Enumeration (CWE) types and includes all necessary information to trigger and validate vulnerability fixes in practice. Each instance we use includes the vulnerable code, its corresponding human fix, and an associated test suite for automated validation.

In our experiments, we use only the PoV (Proof-of-Vulnerability) tests for evaluation. These tests are specifically designed to reproduce the vulnerability, allowing us to directly verify whether a generated patch successfully eliminates the issue. Using PoV tests instead of the full test suites provides a more targeted evaluation focused on vulnerability repair.

2.2 Experimental Setup

All models are prompted using the same predefined prompt configuration, adopted from preliminary publications [2, 1], ensuring a fair comparison by keeping the prompting strategy constant across all evaluated models.

We evaluate three models: GPT-4, GPT-4o, and GPT-5. While the GPT-4 and GPT-4o setups follow previously published configurations, we extend the evaluation by applying the same setup to GPT-5.

The temperature is set to 0.01 to reduce output variability while still allowing minor differences across runs. To account for residual non-determinism, each model is executed three times on the same input sample. This setup allows us to capture variability in model outputs and evaluate both consistency and robustness of the generated fixes.

2.3 Evaluation Procedure

The evaluation process is fully automated and involves multiple steps to ensure consistent and reproducible results. For each generated patch, we first integrate the modified code into the original project by replacing the vulnerable source files. The project is then rebuilt, and the associated PoV tests are executed.

A fix is considered successful if the PoV test, which originally fails on the vulnerable version, passes after applying the generated patch. Test execution results are automatically collected and parsed to determine the outcome of each run.

Model	RUN-1	RUN-2	RUN-3	At least once	Avg. Fixes	Min Fixes	Max Fixes
GPT-4	28% (12)	33% (14)	26% (11)	33% (14)	29% (12.33)	26% (11)	33% (14)
GPT-4o	26% (11)	28% (12)	33% (14)	45% (19)	29% (12.33)	26% (11)	33% (14)
GPT-5	23% (10)	23% (10)	26% (11)	35% (15)	24% (10)	23% (10)	26% (11)

Table 1: Summary of the performance of the evaluated models across the defined metrics.

CWE Category	Total	Fixed
CWE-22	4	3
CWE-611	5	2
CWE-835	7	3

Table 2: Number of correctly repaired vulnerabilities by CWE category (GPT-5, At-least-once).

This process is repeated for all samples and all runs, and the results are aggregated to compute the evaluation metrics. The pipeline ensures consistent handling of code integration, test execution, and result collection across all evaluated models.

3 Results

We evaluated the effectiveness of GPT-5 on repairing 42 Java vulnerabilities from the VUL4J dataset [5], using a fix-prompt strategy to investigate how prompt design influences repair performance.

Each vulnerability instance includes the original faulty code, and an associated test suite, allowing us to systematically measure correctness and reproducibility. By leveraging this curated dataset, which spans multiple CWE categories and contains Proof-of-Vulnerability test cases, we are able to assess GPT-5's ability to generate accurate, context aware fixes in a controlled, real-world setting.

Table 1 summarizes the performance of the evaluated models across the defined metrics. The *At-least-once* metric denotes the number of vulnerabilities for which the model produced at least one correct fix across the three runs. The *Avg. Fixes* represents the average number of correctly fixed vulnerabilities per run, while *Min Fixes* and *Max Fixes* indicate the minimum and maximum number of fixes achieved across the runs, capturing performance variability.

GPT-4 and GPT-4o achieve the highest average number of fixes per run (12.33), while GPT-5 shows at Table 1 a lower average performance (10.33). In terms of the at least once metric, GPT-4o performs best, successfully fixing 19 vulnerabilities in at least one run, compared to 14 for GPT-4 and 15 for GPT-5.

Regarding variability, GPT-5 exhibits more consistent behavior, with a narrow range between minimum and maximum fixes (10-11). In contrast, GPT-4 and GPT-4o show higher variability, with results ranging from 11 to 14 fixes across runs.

These results highlight differences in model behavior rather than identifying a single best model. GPT-4 and GPT-4o tend to achieve higher average numbers of fixes and more successes in at-least-one runs, but their performance is more variable. GPT-5 is more consistent across runs, though its overall repair effectiveness is lower. This illustrates a trade-off between peak effectiveness and stability, showing how different models may behave under identical prompting conditions without implying that one is definitively superior.

A breakdown by Common Weakness Enumeration (CWE) categories (see the examples in Table 2) shows that performance differences persist across vulnerability types, with no single model consistently outperforming the others in all categories. This suggests that newer models

do not necessarily outperform earlier ones in vulnerability repair tasks when using identical prompting strategies.

4 Threats to Validity

Several factors may affect the validity of our results. First, the dataset is limited to 42 Java vulnerabilities from VUL4J, which may not fully represent the diversity of real-world vulnerabilities. Second, we rely on PoV tests for evaluation, which only verify that the known vulnerability is fixed. It is possible that the vulnerability still exists in the code, but the test suite does not detect it, so some side effects or incomplete fixes may go unnoticed. Third, the number of runs per model is limited to three; while sufficient to capture variability trends, more runs could provide a more robust estimate of performance stability. Finally, the evaluation is based on a single prompt configuration, which may influence model behavior and limit the generalizability of results to other prompting strategies.

5 Conclusion

In this study, we compared GPT-4, GPT-4o, and GPT-5 on automatic vulnerability repair using the same prompt configuration. Our results indicate a trade-off between effectiveness and consistency: GPT-4 and GPT-4o achieve higher average and at least once fixes, while GPT-5 exhibits more stable but overall lower repair performance. CWE-based analysis shows that performance varies across vulnerability types, with no single model dominating all categories.

These findings highlight that newer models do not necessarily guarantee better vulnerability repair when using identical prompts. Future work may explore alternative prompting strategies, additional datasets, and increased repetitions to further assess model capabilities and applicability.

References

[1] Z. Ságodi, G. Antal, B. Bogenfürst, M. Isztin, P. Hegedűs, and R. Ferenc, "Reality Check: Assessing GPT-4 in Fixing Real-World Software Vulnerabilities," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE '24)*, pp. 252–261, Association for Computing Machinery, 2024.

[2] G. Antal, B. Bogenfürst, R. Ferenc, and P. Hegedűs, "Identifying Helpful Context for LLM-based Vulnerability Repair: A Preliminary Study," in *Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering (EASE '25)*, pp. 696–700, Association for Computing Machinery, 2025.

[3] MITRE, "Common Weaknesses Enumeration," 2025. [Online]. Available: `https://cwe.mitre.org/`. Accessed: 2025-03-13.

[4] MITRE, "Common Vulnerabilities and Exposures," 2025. [Online]. Available: `https://cve.mitre.org/`. Accessed: 2025-03-13.

[5] Q.-C. Bui, R. Scandariato, and N. E. Díaz Ferreyra, "Vul4J: a dataset of reproducible Java vulnerabilities geared towards the study of program repair techniques," in *Proceedings of the 19th International Conference on Mining Software Repositories*, pp. 464–468, Association for Computing Machinery, 2022.

[6] H. Booth, D. Rike, and G. Witte, "The National Vulnerability Database (NVD): Overview," ITL Bulletin, National Institute of Standards and Technology, Gaithersburg, MD, 2013.

[Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=915172

Taxonomy and Optimization of Kolmogorov-Arnold Networks vs. Multi-Layer Perceptrons for Explainable AI

Moenes Benaissa

Abstract: Multi-layer perceptron (MLP) models are widely used in all sorts of domains and usually adopted as the industry standard, yet they are a black boxes, meaning they lack transparency, which limits their use in critical areas like medicine and finance. To combat this, our paper introduces Kolmogorov-Arnold Network, KAN, as an alternative to the MLP. Shifting the activation function from the nodes to the edges ensures explainability and efficiency. We conducted a thorough literature review and benchmarked KAN variants across multiple datasets and domains. As a result, we found that KAN outperformed MLPs in multiple aspects, and particularly wavelet KAN achieved higher accuracy with fewer parameters in cardiovascular diagnosis.

1 Introduction

Multi-Layer Perceptrons (MLPs) are commonly adopted in a wide range of domains, yet they remain a black boxes, making their internal logic difficult to explain. The combination of affine transformations and fixed non-linear activation functions on nodes causes this issue, hence it limits their full integration in critical fields like finance and medicine. The Kolmogorov-Arnold Network (KAN) introduces a novel mathematical shift, replacing the old fixed node activations with a learnable parametric continuous function on the network's edges. This change results in a reduction in the number of network nodes compared to MLPs and allows a greater parameter density per edge. Consequently, KAN has a smaller architecture, consumes less memory in sparse regimes, and offers mathematical explainability. This research explores whether the theoretical advantages of KAN hold in practice and how to optimize its use.

2 Literature Review

Any multivariate continuous function can be represented as a finite superposition of univariate functions based on the Kolmogorov-Arnold representation theorem, used by Liu et al. [1] Adapting this theorem into neural networks marked an advancement in AI. Surveys highlight KAN's potential across multiple domains [6].

To enhance architectural and computational efficiency, variants including parallelized architectures (MatrixKAN), basis-function approaches (FastKAN, ReLU-KAN), polynomial-based designs (Cheby-KAN), wavelet-based networks (Wav-KAN), and activation-function-based models (AF-KAN) were developed [2, 19, 13, 8, 7, 3].

KANs have also been applied to specialized paradigms: Convolutional KANs for vision, Kolmogorov-Arnold Transformers (KAT) for attention mechanisms, and Kolmogorov-Arnold Graph Neural Networks (KANG) for graph-based learning [5, 17, 11]. Standard temporal KANs and AR-KAN enhance time-series forecasting, while KAN 2.0, KINN, Energy-Dissipative Evolutionary KANs, and KAN-ODEs support physics-informed learning [14, 4, 15, 12, 18, 21]. Variational, Probabilistic, and equivariant KAN models (EKAN) further improve robustness [16, 20, 9]. Finally, KAN has been successfully applied to high-dimensional genomic tasks [10]. Based on the Kolmogorov-Arnold representation theorem, we can apply the formula for a single layer

$$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{1}$$

where $\phi_{q,p}$ are learnable univariate edge functions. In the deep setting, KAN is extended as a multilayer composition:

$$f(\mathbf{x}) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_0)(\mathbf{x}) \tag{2}$$

with layer-wise activations given by

$$\mathbf{x}_j^{(l+1)} = \sum_{i=1}^{n_l} \phi_{l,j,i}(\mathbf{x}_i^{(l)}) \tag{3}$$

3 Experimental Framework and Objectives

3.1 Phase 1: Hardware Performance and Baseline Complexity (Fashion-MNIST)

To isolate basis effects, we evaluated KAN against a parameter-matched 512-node MLP (406,528 parameters) on Fashion-MNIST.

Table 1: Hardware Latency, Architectural Baseline, and Accuracy (CPU)

Architecture	**Width (n_h)**	**Grid (G)**	**Parameters**	**Latency (ms)**	**Overhead**	**Test Acc.**
Placebo MLP	512	N/A	406,528	1.0528	1.00×	87.54%
B-Spline KAN	64	5	406,528	5.2583	4.99×	84.44%

While KAN had a similar parameter count as the MLP, its 3D tensor expansion caused a 4.99× latency penalty. However, dynamic grid extension ($G = 5 \to 15$) increased KAN's accuracy to 86.24%, proving structural refinement boosts capacity.

Table 2: Taxonomy Performance Comparison (Fashion-MNIST - 2 Epoch Sprint)

Variant	**Basis Type**	**Property**	**Test Accuracy**
B-Spline KAN	Spline	Local Support	**86.24%**
Fourier KAN	Trigonometric	Global Periodic	85.80%
Cheby KAN	Orthogonal Poly.	Global Spectral	85.71%
Wav-KAN	Mexican Hat	Multi-Resolution	85.04%

3.2 Phase 2: Burger's Shock Equation Taxonomy

Using the *burgers_shock.mat* dataset, we benchmarked KANs on sharp gradient scientific computing.

Table 3: Real Database Taxonomy: Burger's Shock Performance.

Architecture	**Params**	**MSE**	**Precision**	**Recall**	**Train (s)**	**Inf (ms)**
Placebo MLP	**257**	**0.0288**	**0.7476**	**0.8906**	12.05	0.000338
Efficient-KAN	929	0.0369	0.4391	0.7355	16.05	0.000854
Fast-KAN (RBF)	683	0.0530	0.4688	0.6996	4.82	**0.000191**
Cheby-KAN	673	0.0509	0.4140	0.7195	4.25	0.000220
Fourier-KAN	673	0.0645	0.3889	0.6480	**3.96**	0.000203

The MLP ranked first in accuracy, but Cheby-KAN and Fourier-KAN converged 3× faster, showing an advantage in rapid approximation.

3.3 Phase 3: Feynman Universal Gravitation Benchmark

Assessing symbolic regression potential on $F = G\frac{m_1 m_2}{r^2}$.

Table 4: Real Physics Database: Feynman I.12.1 Results.

Architecture	Params	MSE	Precision	Recall	Train (s)	Inf (ms)
Placebo MLP	321	**0.0211**	**0.9956**	**0.9982**	5.34	0.000394
Cheby-KAN	801	0.3223	0.9296	0.9692	**1.64**	**0.000305**

Cheby-KAN trained faster than MLP while maintaining a strong structural correlation.

3.4 Phase 4: MIT-BIH ECG Clinical Diagnostics

We tested the models on physiological data (1D complex manifold signals).

Table 5: Parameter Efficiency vs. Accuracy on MIT-BIH Dataset.

Architecture	Params	Final Accuracy	V-Class F1
Vanilla MLP (Baseline)	81,669	97.34%	0.94
Pure Wav-KAN	**36,864**	88.85%	**0.88**
Pure Fast-KAN (RBF)	98,304	83.83%	0.69

With 55% fewer parameters, Wav-KAN maintained 91% of the baseline accuracy. By equalizing parameters, accuracy increased to 93.41%, and recall for rare Fusion (Class F) beats improved by +5.0%. Our custom wavelet stabilization reduced inference latency to 0.0983 ms and boosted Class F precision by +172%.

3.5 Implementation and Algorithmic Engineering

We overcome library limitations, by enginnering a custom PyTorch training loops requiring the following key algorithmic interventions:

- **Structural Parameter Parity:** We manually calculated and configured KAN hidden widths (n_h), grid resolutions (G), and spline orders (k) to match the MLP control groups. This required managing custom 3D tensor expansions of shape $[\text{Batch} \times \text{In} \times (G+k)]$ to monitor memory bandwidth bottlenecks.
- **Dynamic Grid Extension:** For spatial datasets, we engineered a we engineered an altering schedule from coarse training to fine training. We initialized KANs at a coarse grid ($G = 5$) and dynamically interpolated to a fine grid ($G = 15$) mid-training, accompanied by manual learning rate decay ($1 \times 10^{-3} \rightarrow 5 \times 10^{-4}$) to increase capacity without adding neurons.
- **Differentiable Wavelet Integration:** For clinical signal processing, initial tests with hard-coded wavelets caused a precision degradation (gradient explosion). To resolve this, we reprogrammed the Wav-KAN layer to utilize standardized, differentiable wavelet transforms with learnable scale (σ) and translation (τ) parameters, stabilized by `BatchNorm1d` and Kaiming initialization.

4 Discussion and Limitations

For recursive spline/basis evaluation, the results show a clear limitation of KANs in computational overhead, as well as notable grid sensitivity. Meanwhile, MLP remains superior for high-dimensional spatial data.

Nonetheless, dynamic grid extension ($G = 5 \rightarrow 15$) proved that a 64-neuron KAN can scale its internal precision to match the MLP's while preserving structural sparsity. The Mexican Hat wavelet in Wav-KAN enabled precise filtering of pathological morphologies, highlighting the need for continuous research into domain-specific inductive biases. This further emphasizes KAN's extraordinary potential.

5 Conclusion

KAN superior parameter efficiency, diagnostic sensitivity, and faster convergence on non-linear symbolic and signal manifolds highlights its great potential. Using the differentiable Wavelet-KAN, it did match MLP accuracy with fewer parameters and ultra-low latency (0.0983 ms). It provides an interpretable, resource-efficient architecture suitable for real-time and edge deployments by optimizing basis functions for specific applications.

References

[1] Z. Liu et al., "KAN: Kolmogorov-Arnold Networks," *arXiv:2404.19756*, 2024.
[2] C. Coffman and L. Chen, "MatrixKAN: Parallelized Kolmogorov-Arnold Network," *arXiv:2502.07176*, 2025.
[3] H.-T. Ta and A. Tran, "AF-KAN: Activation Function-Based KANs," *arXiv:2503.06112*, 2025.
[4] C. Zeng et al., "AR-KAN: Autoregressive-Weight-Enhanced KAN for Time Series," *arXiv:2509.02967*, 2025.
[5] A. D. Bodner et al., "Convolutional Kolmogorov-Arnold Networks," *arXiv:2406.13155*, 2024.
[6] S. Somvanshi et al., "A Survey on Kolmogorov-Arnold Network," *arXiv:2411.06078*, 2024.
[7] Z. Bozorgasl and H. Chen, "Wav-KAN: Wavelet Kolmogorov-Arnold Networks," *arXiv:2405.12832*, 2024.
[8] S. SS et al., "Chebyshev Polynomial-Based Kolmogorov-Arnold Networks," *arXiv:2405.07200*, 2024.
[9] L. Hu et al., "Incorporating Arbitrary Matrix Group Equivariance into KANs," *arXiv:2410.00435*, 2024.
[10] O. Cherednichenko and M. Poptsova, "Kolmogorov-Arnold Networks for Genomic Tasks," *bioRxiv*, 2024.
[11] G. De Carlo et al., "Kolmogorov-Arnold Graph Neural Networks," *arXiv:2406.18354*, 2025.
[12] Y. Wang et al., "Kolmogorov-Arnold-Informed neural network," *arXiv:2406.11045*, 2024.
[13] Q. Qiu et al., "ReLU-KAN: New Kolmogorov-Arnold Networks," *arXiv:2406.02075*, 2024.
[14] K. Xu et al., "Kolmogorov-Arnold Networks for Time Series," *arXiv:2406.02496*, 2024.
[15] Z. Liu et al., "KAN 2.0: Kolmogorov-Arnold Networks Meet Science," *arXiv:2408.10205*, 2024.
[16] F. Alesiani et al., "Variational Kolmogorov-Arnold Network," *arXiv:2507.02466*, 2025.
[17] X. Yang and X. Wang, "Kolmogorov-Arnold Transformer," *arXiv:2409.10594*, 2024.
[18] G. Lin et al., "Energy-Dissipative Evolutionary KANs," *arXiv:2503.01618*, 2025.
[19] Z. Li, "Kolmogorov-Arnold Networks are Radial Basis Function Networks," *arXiv:2405.06721*, 2024.
[20] A. Polar and M. Poluektov, "Probabilistic Kolmogorov-Arnold Network," *arXiv:2104.01714*, 2024.
[21] B. C. Koenig et al., "KAN-ODEs: Kolmogorov-Arnold Network Ordinary Differential Equations," *arXiv:2407.04192*, 2024.

Diffusion-Based Feature Denoising with NNMF for Robust handwritten digit multi-class classification

Hiba Adil Al-kharsan, Róbert Rajkó

Abstract: This work presents a robust multi-class classification framework for handwritten digits that combines diffusion-driven feature denoising with a hybrid feature representation. Inspired by our previous work on brain tumor classification [11], the proposed approach operates in a feature space to improve the robustness to noise and adversarial attacks. This manuscript is submitted as an extended abstract rather than a full-length press-ready paper.

First, the input images are converted into tight, interpretable exemplification using Non-negative Matrix Factorization (NNMF). In parallel, special deep features are extracted using a computational neural network (CNN). These integral features are combined into a united hybrid representation. The main objective of this work is to extend our previously validated two-class framework to a multi-class handwritten digit classification scenario.

To improve robustness, a step diffusion operation is used in the feature space by gradually adding Gaussian noise. A feature denoiser network is trained to reverse this operation and rebuild clean representations from tilted inputs. The courteous features are then applied for multi-class classification.

The suggested method is evaluated in both baseline and adversarial settings using AutoAttack. The experimental outcome present that the diffusion-based hybrid model is both effective and robust, the CNN baseline models outperforming while maintain powerful classification performance. These results explain the activity of feature-level diffusion defense for reliable multi-class handwritten digit classification.

Keywords: handwritten digit classification; multi-class classification; NNMF; diffusion models; feature denoising; AutoAttack; adversarial robustness.

1 Introduction

Handwritten digit multi-class classification is a major problem in machine learning and is widely used to evaluate classification models. Convolutional Neural Networks (CNNs) achieve high accuracy by learning feature representations directly from the data [1]. Although handwritten digit classification is a classical and well-studied task, it is used here as a controlled benchmark to evaluate robustness-oriented feature-level defense mechanisms.

However, deep models remain vulnerable to adversarial perturbations, in which small input modifications can significantly degrade performance [2]. To ensure a reliable evaluation, standardized structures such as AutoAttack are used primarily to estimate robustness [3].

Feature extraction methods such as Non-negative Matrix Factorization (NNMF) provide a consolidated, interpretable representation, complemented by the distinctive power of CNN features [4]. In addition, diffusion-based models have been used to rebuild clean data from noisy information and improve robustness to perturbations [5] NNMF does not directly operate well on mixed-set data; therefore, digit samples are handled in a category-aware manner before constructing the final hybrid representation. Unlike standard handwritten digit classification studies, this work focuses on robustness-oriented feature-space processing and adversarial resistance rather than improving conventional classification accuracy.

In this model, a handwritten digit data set provided as part of the built-in MATLAB examples is used for training and evaluation. This dataset is selected for reproducibility and direct integration within the MATLAB-based implementation pipeline, and serves as a practical proxy similar to MNIST. We propose a hybrid framework that combines NNMF and CNN features with diffusion-based feature denoising to achieve robust multi-class classification. The suggested

method is evaluated in both clean and adversarial tuning, explaining improved accuracy and robustness.

The main contribution of this work is the integration of NNMF-based interpretable features, CNN-based deep features, and diffusion-based feature denoising into a unified pipeline evaluated under adversarial settings. In contrast to our previous work, which focused on a two-class scenario, this study investigates the scalability and behavior of the proposed method in a multi-class setting.

2 Materials and Methods

The suggested method is organized into five main stages: data set preparation, feature extraction, hybrid feature structure and classifier training, diffusion-based feature denoising, and robustness evaluation. These stages are intended to enhance the stability of the classification and the robustness of the adversaries. Due to the limited length of the extended abstract format, only the essential methodological components are summarized.

Preparation of the data set: In this stage, a handwritten digit data set is provided in MATLAB examples [6] and used for training and evaluation. The data set comprises approximately 10,000 grayscale images in 10 classes corresponding to digits (0–9), with almost equal numbers of samples per class. Whole images are re-sized to a steady dimension to guaranty matching for the next processing steps. The data set is collected into three subsets: 70% (7,000 images) for training, 15% (1,500 images) for validation, and 15% (1,500 images) for testing. This stable division ensures that sufficient data is available for learning while maintaining reliable validation during training and unbiased evaluation on unseen samples.

This stage established the foundation of the proposed framework, ensuring that the input data is properly organized, normalized, and prepared for feature extraction and model training in the subsequent stages.

Hybrid Feature Construction Using CNN and NNMF: A CNN is trained in the images to extract special features from a fully connected medium layer, which holds high-level spatial information [7].

In parallel, NNMF uses vectorized images to gain interpretable parts-based representations. The number of components is set to $k = 30$, that is, a summarized feature space while preserving primary structural patterns [8]. The NNMF model is trained on the training set and applied to project the validation and test data. Both CNN and NNMF features are normalized and concatenated to form a united hybrid feature vector. A lightweight neural network multi-class classifier is then trained on the combine features for multi-class classification. This hybrid representation improves performance over individual feature types by merging and interpretable characteristics. The NNMF-based representation follows the principle validated in our previous two-class experiments, here extended to the multi-class case.

Diffusion-Based Feature Denoising: To improve robustness to noise and trouble, a diffusion-based denoising process is used in the feature space. The hybrid feature vectors gained from the previous stage are gradually corrupted by adding Gaussian noise at multiple steps, simulating degraded representations. A denoising network is then trained to reconstruct the original clean features from noisy input by learning the inverse diffusion process. This approach enables the model to recover stable and informative representations even under corrupted conditions. Using a feature space rather than an image space, the proposed method decreases computational complexity while preserving primary special information. The diffusion-based denoising mechanism improves the robustness of the hybrid model and improves the classification performance, following the principles of probabilistic diffusion modeling [5]. The purpose of this stage is to investigate robustness improvement rather than maximizing clean-data accuracy.

Adversarial Evaluation and Performance Analysis. In this stage, the trained data are exported to the ONNX format and evaluated using Python.. The exported classifier and denoising

Case	Accuracy	Precision	Recall	F1	MCC	Balanced Acc	ROC-AUC	LogLoss	Brier
Clean_Base	0.992	0.9921	0.992	0.9920	0.9911	0.992	0.99997	1.4756	0.6605
Clean_Def	0.942	0.9430	0.942	0.9418	0.9357	0.942	0.99827	1.5407	0.6822
Robust_Base	0.507	0.7088	0.507	0.4869	0.4771	0.507	0.97116	1.8536	0.7740
Robust_Def	0.582	0.6333	0.582	0.5751	0.5408	0.582	0.96883	1.7957	0.7580

Table 1: Performance comparison on clean and adversarial data with fixed $k = 30$.

Case	Accuracy	Precision	Recall	F1	MCC	Balanced Acc	ROC-AUC	LogLoss	Brier
Clean_Base	0.9960	0.9961	0.9960	0.9960	0.9956	0.9960	0.99996	1.4693	0.6582
Clean_Def	0.9720	0.9722	0.9720	0.9719	0.9689	0.9720	0.9991	1.5132	0.6734
Robust_Base	0.6920	0.7711	0.6920	0.6864	0.6663	0.6920	0.9912	1.6933	0.7259
Robust_Def	0.7270	0.7966	0.7270	0.7346	0.7047	0.7270	0.9889	1.6817	0.7236

Table 2: Performance comparison on clean and adversarial data after optimized k value for NNMF ($k = 70$).

system are loaded and converted into PyTorch format for active deduction on both CPU and GPU environments.

The estimate is applied to clean and adversarial samples. Firstly, the clean test set is passed through both the baseline classifier and the defended model, which combines the diffusion-based denoising procedure. The defended model executes multiple stochastic forward passes and mediates the predictions to gain a stable outcome.

To estimate robustness, adversarial symbols are created using the AutoAttack framework in the L_∞ threat system with a fixed trouble budget. Specifically, parameter-free attacks such as APGD-CE and APGD-DLR are applied to produce strong adversarial samples [3]. These attacks are designed to provide a reliable and unified evaluation of the system's robustness.

The execution of both baseline and defended data is ranked using multiple metrics, including accuracy, precision, recall, F1-score, Matthews correlation coefficient (MCC), and balanced accuracy. In addition, probabilistic metrics such as ROC-AUC, log-loss, and Brier score are calculated to assess prediction confidence and calibration [9]. A direct comparison with state-of-the-art digit classification models is outside the scope of this extended abstract, as the focus is on robustness evaluation rather than absolute accuracy performance.

The comparison between clean and adversarial results explains that the proposed diffusion-based defense significantly progresses robustness while maintaining competitive performance on baseline data. This emphasizes the efficiency of combining feature-level denoising with hybrid feature representations for reliable classification under adversarial case.

3 Results and Discussion

The results should be interpreted in the context of robustness evaluation and methodological validation. The suggested system is estimated on clean and adversarial test data. Four rating scenarios are considered: the baseline model for clean data, the defended model for clean data, the baseline model on adversarial attack, and the defended model under adversarial attacks. Table 1 shows performance comparison on clean and adversarial data with fixed $k = 30$, and Table 2 shows the same metrics using the optimized $k = 70$ component number for NNMF. We can see a significant improvement on metrics after optimization for the case of Robust_Def.

4 Conclusion

This work represents an initial investigation toward extending the proposed method to multi-class settings, with further evaluation on larger benchmarks planned as future work.

Acknowledgement

This work was supported by the Distinguished Professor Program of Óbuda University. The authors are also grateful for the possibility of using the HUN-REN Cloud https://science-cloud.hu/en [10] which helped us achieve some particular results published in this paper. Hiba Adil Al-kharsan gratefully acknowledges the financial support of the Stipendium Hungaricum Doctoral Program, managed by the Tempus Public Foundation.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015. doi: 10.48550/arXiv.1412.6572. URL: https://doi.org/10.48550/arXiv.1412.6572.

[3] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International Conference on Machine Learning (ICML)*, 2020. doi: 10.48550/arXiv.2003.01690. URL: https://doi.org/10.48550/arXiv.2003.01690.

[4] T. K. Chan, C. S. Chin, and Y. Li, "Non-Negative Matrix Factorization-Convolutional Neural Network (NMF-CNN) for sound event detection," in *Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge*, 2020. URL: https://doi.org/10.48550/arXiv.2001.07874.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin, Eds. Curran Associates, Inc., 2020, vol. 33, pp. 6840–6851.

[6] MathWorks, "Create Simple Deep Learning Network for Classification," 2023. https://www.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html (Accessed: 2026-03-30).

[7] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[8] N. Gillis, *Nonnegative Matrix Factorization*. SIAM, 2020. doi: 10.1137/1.9781611976410.

[9] S. Mazzanti, "It took me 6 years to find the best metric for classification models," *Medium (Data Science Collective)*, 2023. https://medium.com/data-science-collective/it-took-me-6-years-to-find-the-best-metric-for-classification-models-0f5aa21a2b85 (Last accessed: 10.03.2026).

[10] M. Héder, E. Rigó, D. Medgyesi, R. Lovas, S. Tenczer, F. Török, A. Farkas, M. Emődi, J. Kadlecsik, Gy. Mező, Á. Pintér, and P. Kacsuk, "The Past, Present and Future of the ELKH Cloud," *Információs Társadalom*, vol. 22, no. 2, pp. 128–137, 2022. doi: 10.22503/inftars.xxii.2022.2.8.

[11] H. A. Al-kharsan and R. Rajkó, "Diffusion-Based Feature Denoising and Using NNMF for Robust Brain Tumor Classification," *arXiv preprint arXiv:2603.13182*, 2026. URL: https://arxiv.org/abs/2603.13182.

Meteorology-Driven Live Fuel Moisture Content Estimation at Global Scale: A Machine Learning Empirical Study Without Satellite Imagery

Krisztian Pomazi, Bertalan Forstner

Abstract: Live Fuel Moisture Content (LFMC) is a critical variable for wildfire danger assessment, yet most machine learning approaches rely on satellite reflectance data that is unavailable in many operational contexts. We present a systematic empirical study of meteorology-only LFMC estimation at global scale using the Globe-LFMC 2.0 dataset (287,551 field samples from 1,080 sites worldwide). Five models—Ridge regression, a Multi-Layer Perceptron (MLP), Random Forest (RF), XGBoost, and LightGBM—are trained exclusively on 25 features derived from ERA5 reanalysis meteorology, elevation, land cover, and seasonality. Hyperparameters for all tree ensemble models are optimised with Optuna (TPE sampler, 40–50 trials per model) on a site-aware validation holdout. Using a site-aware spatial split to prevent data leakage, the best model achieves $R^2 = 0.572$ and RMSE = 29.0 % on the held-out test set, stable across five random seeds ($R^2 = 0.569 \pm 0.014$). An ablation using only ERA5 meteorological features confirms the atmospheric signal as the primary driver.

Keywords: live fuel moisture content, wildfire, ERA5, machine learning, random forest, vapor pressure deficit, Globe-LFMC, fire weather index

1 Introduction

Live Fuel Moisture Content (LFMC), defined as the ratio of water mass to dry biomass in living vegetation, is a key variable governing fire ignition and spread. Accurate LFMC estimates are essential for wildfire danger rating systems [1] and operational fire management decisions.

Physics-based fire weather indices—notably KBDI [6] and FWI [7]—proxy dead-fuel dryness but do not generalize to live vegetation diversity globally. Satellite MODIS approaches achieve higher accuracy for live fuels [8] but depend on cloud-free overpasses and complex preprocessing chains inaccessible to many fire agencies. ERA5 reanalysis provides global, near-real-time atmospheric variables at 0.25° resolution [2], offering a practical satellite-free alternative. Note that ERA5 itself assimilates spaceborne radiances, so this approach is not entirely satellite-independent.

Prior LFMC machine learning has relied largely on satellite reflectance, from deep learning on MODIS time series [10] and multi-modal CNNs fusing reflectance with meteorology [11], to optical–SAR recurrent models [12]; purely meteorological prediction has so far been confined to regional drought-index studies [13]. The release of Globe-LFMC 2.0 [1]—the largest harmonized global LFMC dataset with pre-matched AgERA5 variables—enables a large-scale benchmark for the first time. This paper contributes:

- A 5-model ML empirical study (including an MLP neural baseline) on 287,551 globally distributed samples with site-aware spatial evaluation.
- Physically-motivated derived features (VPD, antecedent precipitation ratio) and a geographic ablation separating the ERA5 signal from location proxies.
- Optuna hyperparameter optimisation for all tree ensemble models on a site-aware validation holdout, with no leakage from the test set.
- Split stability verified across five random seeds (R^2 variance <0.01).
- Error analysis by IGBP land cover type to guide vegetation-specific modelling.

Globe-LFMC 2.0 is dominated by North America, Australia, and Mediterranean Europe; performance on underrepresented continents warrants targeted future study.

2 Data and Methods

2.1 Dataset

Globe-LFMC 2.0 [1] aggregates field-collected LFMC measurements from 1,080 sites across six continents (293,796 records) with pre-matched AgERA5 atmospheric variables. We removed 6,245 rows (≈2 %) flagged by the built-in Isolation Forest quality filter, retaining **287,551 samples**. The target spans 0–1,279 % (global mean 105.8 %).

2.2 Feature Engineering

ERA5 meteorological inputs (14): Precipitation over five windows (24 h, 3 d, 1 week, 4 weeks, 12 weeks); relative humidity at 06:00, 09:00, 12:00, 15:00 UTC; daily T_{max} and T_{mean}; vapour pressure; wind speed; dewpoint.

Derived features (2): VPD ($6.112\, e^{17.67T_c/(T_c+243.5)} - e_a$, August-Roche-Magnus, where T_c is daily mean 2 m temperature in °C and e_a is actual vapour pressure in hPa) captures atmospheric water demand; precipitation ratio $P_{1\text{wk}}/(P_{12\text{wk}}+1)$ captures short-term drought intensity relative to the long-term background.

Contextual features (9): Elevation; IGBP land cover ID; species functional type (the plant functional-type class supplied in Globe-LFMC 2.0, encoded as an integer); latitude; longitude; $\sin/\cos(2\pi\,\text{DOY}/365)$; calendar year (captures climatological trends without temporal leakage since split is by site).

In total **25 features** are used, none requiring real-time satellite access.

2.3 Train/Test Split and Hyperparameter Optimisation

A site-aware split assigns 80 % of sites to training and 20 % to testing (seed 42; 57,510 test samples), so no site is shared between partitions; nearby sites may still straddle the split, so the coordinate features reflect spatial interpolation rather than per-site memorisation. RF R^2 across five random seeds (42, 0, 1, 2, 3) was $\mathbf{0.569 \pm 0.014}$, confirming robustness to the particular partition.

Hyperparameters for RF, XGBoost, and LightGBM were optimised with **Optuna** [9] (TPE sampler) using a second site-aware validation holdout (20 % of training sites, seed 99). The test set was never accessed during search. RF used 40 trials; XGBoost and LightGBM used 50 trials each with early stopping (patience 50) to determine optimal tree counts efficiently. Ridge (α) and MLP were not tuned: grid search showed Ridge RMSE is insensitive to α (confirming non-linearity), and MLP tuning was deferred as tree ensembles are better suited to this tabular structure.

2.4 Models

Five regression models were trained:

Ridge: L_2 ($\alpha = 1.0$), StandardScaler inputs (linear baseline).

MLP: three hidden layers (256–128–64), ReLU, early stopping (patience 20), StandardScaler inputs.

RF [3] *(tuned):* 500 trees, max_depth = 27, min_samples_leaf = 2, max_features = 0.5. Tuning produced minimal change from the defaults, confirming RF is robust to hyperparameter choice on this dataset.

XGBoost [9] *(tuned):* 1,409 estimators, lr = 0.014, depth = 10, min_child_weight = 14, subsample = 0.63, colsample_bytree = 0.80, $\alpha_{\text{L1}} = 0.79$, $\lambda_{\text{L2}} = 1.89$. The tuner moved towards deeper trees (10 vs 7) strongly regularised by a high min_child_weight, compensating complexity with aggressive L1 penalty.

LightGBM [5] *(tuned):* 1,813 estimators, lr = 0.011, num_leaves = 243, min_child_samples = 10, subsample = 0.82, colsample_bytree = 0.81, $\alpha_{L1} = 3.39$, $\lambda_{L2} = 0.39$. The most dramatic change: num_leaves increased from 63 to 243 (4× more expressive trees), offset by a 34× increase in L1 regularisation and a 3× slower learning rate. The default 63 leaves was severely underfitting this 287k-sample dataset.

3 Results and Discussion

3.1 Model Performance

Table 1 reports test-set metrics (57,510 samples from unseen sites). **RF remains the best model after tuning** ($R^2 = 0.572$, RMSE = 29.0 %, $r = 0.757$), essentially unchanged from its pre-tuning baseline—confirming that RF defaults are already near-optimal for this dataset. **LightGBM shows the largest gain from tuning**: RMSE improved from 29.69 to 29.40 and R^2 from 0.552 to 0.561, driven by the 4× increase in num_leaves ($63 \rightarrow 243$). Despite this improvement, LightGBM does not surpass RF, suggesting that additional capacity alone is insufficient and that performance is partly bottlenecked by the feature set rather than model complexity. XGBoost improves only marginally (RMSE $31.15 \rightarrow 31.04$), indicating its baseline configuration was already close to the optimum within the gradient-boosting family. Ridge and MLP both perform poorly ($R^2 < 0.08$), confirming that LFMC–meteorology relationships are strongly non-linear. Field LFMC carries 5–10 % sampling uncertainty; residual model error reflects species and spatial diversity rather than sensor noise.

The ablation RF† (ERA5-derived features only, no geographic proxies) achieves $R^2 = 0.318$, well above the linear baseline (0.079), confirming genuine ERA5 predictive power. The gap to 0.572 quantifies the contribution of geographic and vegetation-type context. For reference, MODIS satellite methods on regional datasets achieve $R^2 = 0.55$–0.80 [8]; the best tuned model (RF, $R^2 = 0.572$) sits at the lower end of this range using meteorology alone.

Table 1: Test-set performance on Globe-LFMC 2.0 (site-aware 80/20 split, 57,510 samples). Bold: best per metric. †ERA5-based features (14 raw + VPD + precipitation ratio); excludes all geographic, land-cover, and vegetation context. Tree ensemble models use Optuna-tuned hyperparameters.

Model	RMSE (%)	MAE (%)	R^2	r	Bias (%)
Ridge	42.58	28.60	0.079	0.321	3.24
MLP	42.91	24.25	0.065	0.549	2.80
XGBoost (tuned)	31.04	21.05	0.511	0.721	1.25
LightGBM (tuned)	29.40	20.68	0.561	0.749	0.67
RF† (ERA5-derived)	36.64	25.02	0.318	0.568	2.98
RF (tuned, all 25 feat.)	**29.02**	**20.26**	**0.572**	**0.757**	1.29

3.2 Feature Importance and Predictions

The top-15 RF feature importances (mean decrease in impurity) are led by species functional type (17.4 %), 12-week antecedent precipitation (14.7 %), longitude (12.8 %), latitude (10.0 %), and elevation (7.8 %). Geographic and vegetation-type proxies thus dominate, consistent with the ablation result: removing them drops R^2 from 0.572 to 0.318. Among purely meteorological variables, 4-week precipitation (5.4 %), mean temperature (3.0 %), wind speed (2.8 %), and VPD (2.4 %) follow, with relative humidity channels clustered near 2.1 % each. Impurity-based

importance systematically overestimates continuous high-cardinality features (coordinates); SHAP values would likely elevate VPD relative to geographic variables.

Fig. 1 shows observed vs. predicted LFMC. The model performs well for 50–200 % but degrades at extreme values (>300 %) corresponding to hygrophilous species underrepresented in training. Error by IGBP land cover reveals higher variability in open shrublands and savannas vs. closed forests, motivating vegetation-specific models.

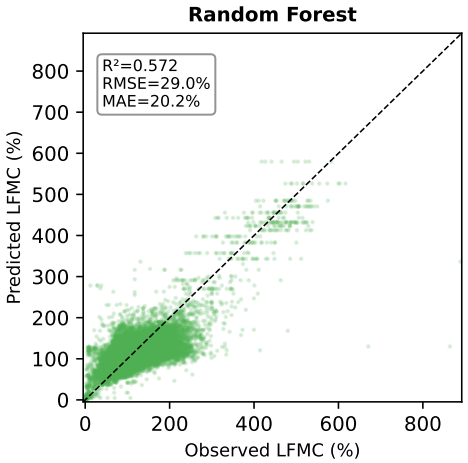


Figure 1: Observed vs. predicted LFMC — best tuned model. Dashed line: 1:1 reference.

4 Conclusion

ERA5 meteorological reanalysis enables LFMC estimation at global scale without user-side satellite processing, with RF achieving $R^2 = 0.572$ and RMSE = 29.0 % after Optuna tuning. Optimisation confirmed that RF defaults are near-optimal, while LightGBM benefited most from tuning (RMSE 29.69 → 29.40) by increasing tree complexity (num_leaves 63 → 243) paired with strong L1 regularisation. That RF still leads despite LightGBM's gains indicates that the performance ceiling is driven by the meteorological feature set rather than model capacity. Among meteorological variables, long-term antecedent precipitation and VPD are the leading drivers; geographic and species-type proxies contribute substantially, as confirmed by the ablation (R^2 drop from 0.572 to 0.318 without them). Fire agencies with weather station or reanalysis access can deploy this approach without satellite pipelines. Future work should pursue MODIS fusion, vegetation-specific predictors, and recurrent networks (LSTM) exploiting temporal sequence structure.

AI Disclosure

The preparation of this manuscript involved the use of Claude.ai for linguistic editing and for assistance with writing and debugging data analysis code. All AI-generated or AI-assisted content was subsequently reviewed, verified, and revised by the authors, who assume full responsibility for the accuracy and integrity of the reported work.

References

[1] M. Yebra et al., "Globe-LFMC 2.0, an enhanced and updated dataset for live fuel moisture content research," *Scientific Data*, vol. 11, no. 1, p. 332, 2024.

[2] H. Hersbach et al., "ERA5 hourly data on single levels from 1940 to present," *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*, 2020.

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[5] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[6] J. J. Keetch and G. M. Byram, "A drought index for forest fire control," USDA Forest Service Research Paper SE-38, 1968.

[7] C. E. Van Wagner, "Development and structure of the Canadian forest fire weather index system," Canadian Forestry Service Forestry Technical Report 35, 1987.

[8] M. Yebra et al., "A global review of remote sensing of live fuel moisture content for fire danger assessment: Moving towards operational products," *Remote Sensing of Environment*, vol. 136, pp. 455–468, 2013.

[9] T. Akiba et al., "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[10] L. Zhu et al., "Live fuel moisture content estimation from MODIS: A deep learning approach," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 179, pp. 81–91, 2021.

[11] L. Miller et al., "Multi-modal temporal CNNs for live fuel moisture content estimation," *Environmental Modelling and Software*, vol. 156, p. 105467, 2022.

[12] K. Rao et al., "SAR-enhanced mapping of live fuel moisture content," *Remote Sensing of Environment*, vol. 245, p. 111797, 2020.

[13] J. Ruffault et al., "How well do meteorological drought indices predict live fuel moisture content (LFMC)? An assessment for wildfire research and operations in Mediterranean ecosystems," *Agricultural and Forest Meteorology*, vol. 262, pp. 391–401, 2018.

Evaluating LLMs for SAST Warning Repair: An Empirical Study on VUL4J

István Kolláth, Gábor Antal

Abstract: Static Application Security Testing (SAST) tools are widely adopted in modern software development pipelines to identify potentially security-relevant code patterns. However, such tools typically generate a substantial volume of warnings, making their manual repair time-consuming and prone to human error. In this work, we investigate the use of medium-sized open-weight Large Language Models (LLMs) for the automated repair of SAST warnings. Unlike proprietary LLMs, medium-sized open-weight LLMs can be deployed locally, offering a more privacy-preserving and cost-effective alternative. We evaluate multiple LLM families, including models from the Qwen and GPT open-source families, comparing their ability to generate candidate patches for selected SAST warnings. In our experiment, we select 22 warnings from the VUL4J real-world dataset. Our results indicate that `Qwen3.5-35B` achieves the best overall performance with a FixRate of **63.64%** and **82** successful repairs out of 220 attempts, while combining all evaluated models raises the FixRate to **77.27%**. Overall, our findings suggest that performance differences among the evaluated LLMs are meaningful in automated SAST warning repair, and that model selection can influence repair effectiveness.

Keywords: static analysis, SAST, warning repair, large language models, automated program repair

1 Introduction

Static Application Security Testing (SAST) tools are now essential for automated fixing, allowing developers to find vulnerabilities and warnings in the source code without executing the program [1]. **SpotBugs** [2], the successor to the well-known **FindBugs** analyzer, is one of the most widely used open-source Java SAST tools. SpotBugs uses pattern-based analysis to detect various issues and integrates easily into Maven and Gradle build pipelines [3].

Recent advances in **Large Language Models (LLMs)** offer new possibilities for automating these repairs. However, large proprietary models such as `GPT-4` come with high costs and require sending potentially sensitive source code to external servers. This paper evaluates and compares multiple LLM families - including models from the Qwen and GPT-OSS families - on their ability to fix SpotBugs warnings. We structure our investigation around two key research questions:

RQ1: *Which performs better at fixing SAST warnings, Qwen or GPT-OSS?*

RQ2: *Can combining multiple LLM families improve the overall repair rate?*

2 Related Work

2.1 SAST Tools and Their Limitations

Static analysis tools are increasingly being used to improve software quality and security. Johnson et al. [4] conducted a well-regarded study based on interviews with 20 developers, revealing that while developers generally recognize the advantages of SAST tools, high false positive rates and low-quality tool output remain the primary barriers to their adoption. Ami et al. [1] further examined industry perspectives through practitioner interviews, revealing that false negatives - vulnerabilities missed by the tool - are considered even more dangerous than false positives, as they create an unwarranted sense of security.

SpotBugs [2], the successor to FindBugs, is one of the most widely used open-source SAST tools for Java. Liu et al. [3] conducted a comprehensive comparison of six Java quality assurance tools and found that SpotBugs successfully detects bugs in Java source codes.

2.2 LLMs for Automated Vulnerability Repair

The use of LLMs for automated program repair (APR) has grown rapidly in recent years. Xia and Zhang [5] demonstrated that conversational interaction with ChatGPT can fix a substantial number of bugs at minimal cost, highlighting the practical potential of LLM-based APR without task-specific fine-tuning. Wadhwa et al. [8] proposed CORE, a tool that uses GPT-3.5 and GPT-4 to fix warnings flagged by static analysis tools. CORE achieved a FixRate of **76.8%** on Java files, demonstrating that LLMs can effectively repair SAST warnings with minimal engineering effort. More recently, CodeCureAgent [9] introduced an agentic approach that autonomously classifies warnings as true or false positives before attempting a repair, achieving a correct-FixRate of **86.3%** on SonarQube warnings.

3 Methodology

3.1 Model Selection

Previous work on LLM-assisted code repair, such as CORE [8], relies on proprietary LLMs like GPT-4, which have high costs and require sending potentially sensitive source code to external servers. To address these practical concerns, we focus on medium-sized open-weight models that can be deployed locally, offering a more privacy-preserving and cost-effective alternative. We evaluated four models from two families. From the **GPT-OSS** family, we select gpt-oss-20b and gpt-oss-20b-safeguard. From the **Qwen** family, we select Qwen3.5-35B-A3B and Qwen3.5-9B, both of which are widely used on HuggingFace and represent comparable parameter scales.

3.2 Dataset

We use the **VUL4J** dataset [10], a collection of real-world reproducible Java vulnerabilities designed for the study of program repair techniques. While the dataset primarily contains vulnerability entries, it also includes 50 SAST warning entries produced by SpotBugs. From this set, we selected warnings that could be fixed in a single file and a single method, resulting in a dataset of **22 warnings**.

3.3 Repair Procedure

For each of the 22 warnings, we created a custom zero-shot prompt containing the SpotBugs warning message and the relevant source code snippet. Each model was run 10 times per warning to generate candidate patches, resulting in a total of 220 patches per model. All models were run locally on an NVIDIA H100 GPU.

3.4 Evaluation

We evaluate the generated patches by re-running SpotBugs on the patched code and checking whether the original warning is still present. A repair is considered successful if the targeted warning disappears without new SpotBugs warnings. We used two metrics: **Fix@10 Count**, the number of warnings for which at least one successful repair was generated across 10 runs, and **FixRate**, the percentage of the 22 warnings for which at least one successful repair was produced.

ID	gpt-oss-20b	gpt-oss-20b-s	Qwen-35B	Qwen-9B	Fix@10
VUL4J-83-S	0	0	0	0	×
VUL4J-85-S	0	0	0	0	×
VUL4J-87-S	10	10	6	0	✓
VUL4J-89-S	10	10	10	3	✓
VUL4J-90-S	2	6	10	9	✓
VUL4J-92-S	1	8	7	1	✓
VUL4J-93-S	0	2	0	3	✓
VUL4J-95-S	1	7	9	7	✓
VUL4J-97-S	1	0	1	0	✓
VUL4J-100-S	9	5	5	9	✓
VUL4J-102-S	0	1	0	0	✓
VUL4J-103-S	0	0	0	0	×
VUL4J-104-S	4	3	6	0	✓
VUL4J-105-S	0	0	0	0	×
VUL4J-106-S	0	0	7	0	✓
VUL4J-108-S	0	2	9	7	✓
VUL4J-109-S	10	10	1	10	✓
VUL4J-110-S	8	1	1	3	✓
VUL4J-111-S	6	8	0	2	✓
VUL4J-114-S	10	3	4	3	✓
VUL4J-115-S	0	0	0	0	×
VUL4J-119-S	0	0	6	6	✓
Fix@10 Count	**12**	**14**	**14**	**12**	**17**
FixRate	**54.55%**	**63.64%**	**63.64%**	**54.55%**	**77.27%**

Table 1: Successful repairs per model (out of 10 runs)

4 Results

4.1 RQ1: Which performs better at fixing SAST warnings, Qwen or GPT-OSS?

Table 1 presents the number of successful repairs generated by each model across the 22 selected SpotBugs warnings. In terms of FixRate, `gpt-oss-20b-safeguard` and `Qwen3.5-35B` achieve the highest individual FixRate of **63.64%** (14 out of 22 warnings fixed at least once), followed by `gpt-oss-20b` and `Qwen3.5-9B`, both at **54.55%** (12 out of 22). Considering the total number of successful repairs across all 220 attempts per model (22 warnings × 10 runs), `Qwen3.5-35B` leads with **82** successful repairs, followed by `gpt-oss-20b-safeguard` with **76**, `gpt-oss-20b` with **72**, and `Qwen3.5-9B` with **63**. `Qwen3.5-35B` thus performs best across both metrics, achieving the highest FixRate jointly with `gpt-oss-20b-safeguard` while also generating the most successful repairs in absolute terms. This suggests that `Qwen3.5-35B` produces more correct fixes across repeated runs, making it the strongest individual model in our evaluation, and this indicates that the Qwen family outperforms the GPT-OSS family in this setting.

4.2 RQ2: Can combining multiple LLM families improve the overall repair rate?

To investigate whether combining the outputs of multiple models is beneficial, we compute the Fix@10 rate across all models jointly - that is, a warning is considered fixed if at least one of the four models produced a successful repair in at least one of its ten runs. The combined FixRate reaches **77.27%** (17 out of 22 warnings), which substantially exceeds the FixRate of any individual model. This suggests that the models exhibit complementary strengths, and that combining outputs from multiple LLM families is a promising strategy for improving the overall

coverage of automated SAST warning repair.

5 Conclusion

In this paper, we presented a preliminary evaluation of medium-sized open-weight LLMs for automated SAST warning repair on SpotBugs findings from the VUL4J dataset. We compared four models from two families using a zero-shot prompting strategy across 22 warnings. Our results show that `Qwen3.5-35B` achieves the best overall performance, leading in both FixRate and total number of successful repairs. However, combining the output of all models raises the FixRate to **77.27%**, substantially exceeding any individual model, suggesting that model combination is a promising direction to improve repair coverage. In future works, we will extend the evaluation to a larger set of warnings and investigate other model families, such as Llama and Mistral, to provide a more comprehensive comparison of open-weight LLMs for SAST warning repair.

References

[1] A. S. Ami, K. Moran, D. Poshyvanyk, and A. Nadkarni, "'False negative – that one is going to kill you': Understanding Industry Perspectives of Static Analysis Based Security Testing," in *Proceedings of the 2024 IEEE Symposium on Security and Privacy (S&P)*, 2024.

[2] SpotBugs Development Team, "SpotBugs: Find Bugs in Java Programs," `https://spotbugs.github.io/`, accessed March 30, 2026.

[3] H. Liu, S. Chen, R. Feng, C. Liu, K. Li, Z. Xu, L. Nie, Y. Liu, and Y. Chen, "A Comprehensive Study on Quality Assurance Tools for Java," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pp. 285–297, 2023.

[4] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why Don't Software Developers Use Static Analysis Tools to Find Bugs?" in *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, pp. 672–681, 2013.

[5] C. S. Xia and L. Zhang, "Keep the Conversation Going: Fixing 162 out of 337 Bugs for $0.42 Each Using ChatGPT," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pp. 819–831, 2024.

[6] U. Kulsum, H. Zhu, B. Xu, and M. d'Amorim, "A Case Study of LLM for Automated Vulnerability Repair: Assessing Impact of Reasoning and Patch Validation Feedback," in *Proceedings of the 1st ACM International Conference on AI-Powered Software (AIware)*, 2024.

[7] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining Zero-Shot Vulnerability Repair with Large Language Models," in *Proceedings of the 44th IEEE Symposium on Security and Privacy (SP)*, pp. 2339–2356, 2023.

[8] N. Wadhwa et al., "CORE: Resolving Code Quality Issues Using LLMs," *Proceedings of the ACM on Software Engineering*, vol. 1, pp. 789–811, 2024.

[9] P. Joos, I. Bouzenia, and M. Pradel, "CodeCureAgent: Automatic Classification and Repair of Static Analysis Warnings," *arXiv preprint arXiv:2509.11787*, 2025.

[10] Bui, Quang-Cuong and Scandariato, Riccardo and Ferreyra, Nicolás E. Díaz, "Vul4J: A Dataset of Reproducible Java Vulnerabilities Geared Towards the Study of Program Repair Techniques," in *Proceedings of the 19th IEEE/ACM International Conference on Mining Software Repositories (MSR)*, pp. 464–468, 2022.

Passive Detection of Forward Error Correction Schemes from Network Traffic

Adam Domonkos, Peter Ekler

Abstract: Passively identifying the active forward error correction (FEC) scheme from encrypted network flows is useful for network management and security auditing, but no labelled real-world dataset exists for this task. A synthetic traffic generator is presented covering four classes (RLNC, Reed-Solomon, Fountain codes, and uncoded UDP) under varied application profiles, lossy channel conditions, and four encryption modes, yielding 56 330 training and 1 000 test samples. Three classifiers are evaluated: a CNN on raw packet bytes (62.2% accuracy, AUC = 0.871), an MLP over 46 engineered features (81.1%, AUC = 0.948), and XGBoost on the same features (97.9%, AUC = 0.9995). SHAP analysis shows that packet-count, size, and timing statistics are the strongest discriminating signals even under full payload encryption.

Keywords: network coding, RLNC, FEC detection, traffic classification, ML, synthetic data generation

1 Introduction

FEC schemes add redundancy so that receivers can reconstruct lost packets without retransmission. Reed-Solomon [4] operates on fixed-size blocks, while rateless LT/Fountain codes [5] generate encoded symbols until the receiver accumulates enough to decode. RLNC [1, 2, 3] allows every forwarding node to linearly combine packets over a finite field (GF(2^8) in this work), boosting throughput on multi-path and broadcast topologies.

Passively identifying which scheme is active is relevant for QoS monitoring, security auditing, and FEC adoption studies [7]. Modern TLS renders payload bytes uniformly high-entropy, so only structural header, size, and timing patterns remain observable. No public labelled PCAP dataset for FEC classification exists, which motivates a synthetic generation approach.

This paper presents a synthetic PCAP generator covering four FEC classes with configurable application profiles, channel impairments, and encryption. Three classifiers are trained and compared: a CNN operating on raw bytes, and XGBoost and MLP models over 46 interpretable features. SHAP attribution is used to identify which structural signals survive payload encryption.

2 Related Work

Traffic classification surveys [7] and deep-learning classifiers on raw encrypted bytes [8] form the basis of the CNN approach used here. RLNC was established by Ahlswede et al. [1], formalised algebraically by Koetter and Médard [2], and made practically random-linear by Ho et al. [3]. Passive ML-based FEC identification from encrypted traffic without node cooperation has not been systematically studied in prior work. XGBoost [9] and SHAP [10] are used for the feature-based models and interpretability analysis.

3 Synthetic Traffic Dataset

3.1 Coding Schemes

All four classes are implemented from scratch to guarantee exact ground-truth labels. RLNC uses a custom GF(2^8) encoder supporting both systematic and non-systematic modes with multi-generation transfers for large flows. In systematic mode the original packets are sent first,

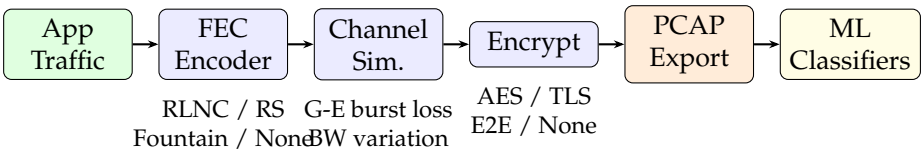


Figure 1: Data generation pipeline. Application traffic is encoded with one of four FEC schemes, passed through a configurable lossy channel, encrypted, and exported as PCAP files for classifier

followed by linearly coded repair symbols, which produces a noticeable shift in the payload byte entropy profile. Reed-Solomon applies systematic erasure coding where the generator matrix has Vandermonde structure and is converted to systematic form $[I_k \mid P]$ via Gaussian elimination, resulting in characteristic packet-count and size patterns tied to block boundaries. Fountain/LT codes use rateless encoding with a Robust Soliton degree distribution [5]. The sender transmits until an ACK confirms decoding, giving variable flow lengths and a distinct burst structure. The uncoded class transmits plain UDP datagrams without any FEC layer.

3.2 Traffic Pattern and Channel Simulation

Each trace blends five application profiles (video streaming, VoIP, real-time video, text messaging, bulk transfer) with independent packet-size and IAT statistics.

Channel impairments follow a Gilbert-Elliott burst loss model [6] with per-flow parameters ($p_G \approx 0$, $p_B = 0.3$–0.9, transition probabilities p_{GB}, p_{BG}) configured to match wired (<0.1%), Wi-Fi (0.5–8%), and cellular (5–15%) loss rates. Bandwidth varies via a Gaussian random walk or ON/OFF model.

Encryption is applied after FEC coding. Each flow is assigned one of four modes: unencrypted (20%), AES (30%), TLS (40%), or E2E (10%), with per-flow random keys. This makes payload bytes high-entropy for all classes, so classifiers must rely on structural header and timing patterns rather than payload content.

The final dataset contains 56 330 training and 1 000 test samples, approximately balanced across four classes, exported as single-flow PCAP files. Figure 1 illustrates the generation pipeline.

4 Classification Models

4.1 Model 1: CNN on Raw Packet Bytes

Each PCAP is converted to a $2\times64\times256$ float32 tensor where channel 0 holds the first 256 bytes of each of the first 64 packets normalised to $[0, 1]$, and channel 1 is a binary direction flag (0 = client-to-server, 1 = server-to-client) replicated across all 256 positions of the packet row. Shorter flows are zero-padded.

The CNN (Fig. 2, left) consists of four convolutional blocks, each with batch normalisation and ReLU, followed by global average pooling. The first two blocks use asymmetric 3×5 kernels to capture within-packet byte patterns along the byte axis (width = 5) and cross-packet burst structure along the packet axis (height = 3). The final two blocks use 3×3 kernels for higher-level features. A fully-connected layer (FC $256 \to 128$) with 0.3 dropout feeds the Softmax output, trained with AdamW and cosine annealing over 30 epochs.

4.2 Model 2: Engineered Features — XGBoost and MLP

46 features are extracted per PCAP across volume, packet size, IAT, bidirectional ratio, payload entropy, burst structure, size distribution, and session statistics.

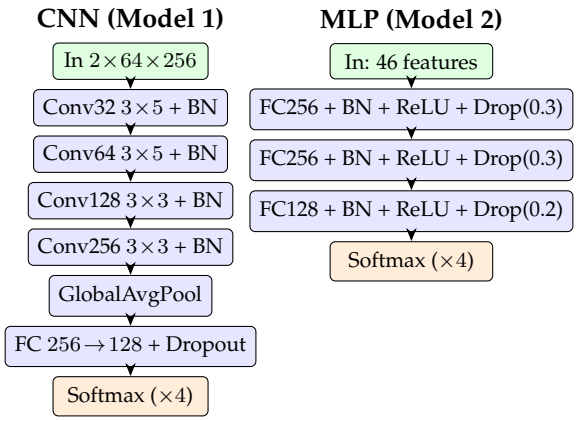


Figure 2: CNN (Model 1) and MLP (Model 2) architectures. XGBoost uses the same 46-feature vector as the MLP without a neural network.

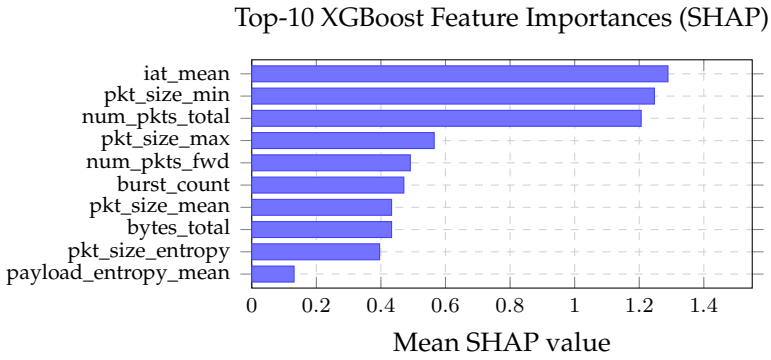


Figure 3: Mean absolute SHAP values for XGBoost (descending order). Timing, packet-count, and size features are the strongest predictors.

The most discriminative features come from packet-count, size, and timing statistics. RLNC flows tend to carry more packets per session (`num_pkts_total`) due to coded repair symbols, and the minimum packet size (`pkt_size_min`) is elevated in systematic mode. Fountain codes produce variable flow lengths reflected in the burst and session features. All features are standardised (zero mean, unit variance) before training. SHAP values are computed via TreeExplainer on the held-out test set after inverse-standardising to original units. As shown in Fig. 3, `iat_mean`, `pkt_size_min`, and `num_pkts_total` are the three highest-importance features by mean absolute SHAP value.

XGBoost [9] is trained with 500 trees, depth 6, and 5-fold CV with a final refit on the full training set. The MLP ($46 \to 256 \to 256 \to 128 \to 4$, BN, dropout 0.3/0.3/0.2, AdamW, 50 epochs) is trained on the same feature vector.

4.3 Results

Table 1 shows performance on the 1 000-sample held-out test set. XGBoost reaches 97.9% accuracy (AUC = 0.9995) with perfect per-class F1 for both RLNC and Fountain. This likely reflects the clean feature distributions of the synthetic dataset and may not transfer directly to real-world traffic. The MLP reaches 81.1% (AUC = 0.948), performing well on the uncoded class (F1 = 0.961) but with lower accuracy on RLNC (F1 = 0.662). The CNN reaches 62.2% (AUC = 0.871) without any feature engineering. Its main weakness is RLNC detection (recall 25.8%), since coded repair symbols are hard to distinguish from payload bytes under encryption. In all three cases, full payload encryption does not prevent FEC identification, as inter-packet statistics and

size distributions carry enough discriminating information.

				Per-class F1			
Model	Acc.	Macro F1	AUC	RLNC	RS	Fountain	None
CNN	0.622	0.603	0.871	0.406	0.964	0.582	0.459
MLP	0.811	0.804	0.948	0.662	0.726	0.867	0.961
XGBoost	**0.979**	**0.979**	**0.9995**	**1.000**	**0.959**	**1.000**	**0.956**

Table 1: Test-set performance on the held-out test set ($n = 1\,000$). AUC is the macro one-vs-rest AUROC. Bold indicates best performance per column.

5 Conclusion

The results show that FEC schemes are passively detectable from encrypted flows using packet-count, size, and timing statistics. XGBoost reaches 97.9% accuracy, with `iat_mean`, `pkt_size_min`, and `num_pkts_total` identified as the most important features. These signals reflect structural differences between coding schemes that remain observable under full payload encryption. The CNN achieves 62.2% without feature engineering, though the feature-based models perform considerably better. Future work will focus on validation with live traffic and extending the generator to batched RLNC variants and additional FEC schemes.

Acknowledgements

The work presented in this paper has been carried out in the frame of project no.2024-1.1.1-KKV_FÓKUSZ-2024-00039, which has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2024-1.1.1-KKV_FÓKUSZ funding scheme.

References

[1] R. Ahlswede et al., "Network information flow", *IEEE Trans. Inf. Theory*, 46(4):1204–1216, 2000.

[2] R. Koetter, M. Médard, "An algebraic approach to network coding", *IEEE/ACM Trans. Netw.*, 11(5):782–795, 2003.

[3] T. Ho et al., "A random linear network coding approach to multicast", *IEEE Trans. Inf. Theory*, 52(10):4413–4430, 2006.

[4] I. S. Reed, G. Solomon, "Polynomial codes over certain finite fields", *J. SIAM*, 8(2):300–304, 1960.

[5] M. Luby, "LT codes", *Proc. 43rd IEEE FOCS*, pp. 271–280, 2002.

[6] E. N. Gilbert, "Capacity of a burst-noise channel", *Bell Syst. Tech. J.*, 39(5):1253–1265, 1960.

[7] T. T. T. Nguyen, G. Armitage, "A survey of techniques for internet traffic classification using ML", *IEEE Commun. Surv. Tutor.*, 10(4):56–76, 2008.

[8] M. Lotfollahi et al., "Deep packet: Encrypted traffic classification using deep learning", *Soft Comput.*, 24(3):1999–2012, 2020.

[9] T. Chen, C. Guestrin, "XGBoost: A scalable tree boosting system", *Proc. ACM SIGKDD*, pp. 785–794, 2016.

[10] S. M. Lundberg, S.-I. Lee, "A unified approach to interpreting model predictions", *Advances NeurIPS*, pp. 4765–4774, 2017.

Point-Curve Minimal Distance via Higher-Order Taylor Estimation of Footpoint Mapping

Viktor Vad, Gábor Valasek

1 Problem Statement

Determining the minimal distance between a given input curve and a query point is crucial in various applications, including computer graphics and geometric design. For Bézier curves, specifically tailored methods leverage the divide-and-conquer paradigm to efficiently compute distances by breaking the problem into manageable subproblems and solving them recursively[1]. Despite the efficiency and robustness of these methods, their applicability is tied to polynomial curves.

For general parametric curves, the standard method of computing closest points involves using the derivative of the squared distance function, which has zeros at parameters where the tangent of the curve is orthogonal to the vector from the point to the curve. Since this yields a nonlinear problem, root-finding techniques are typically applied using iterative numerical methods such as Newton–Raphson or geometric Newton iteration[2, 3, 4, 6]. Although these methods generally exhibit a second-order convergence rate in the neighborhood of roots, ensuring accurate results for a specific query point can be challenging because iterative methods are sensitive to their starting points. Furthermore, the roots can indicate tangent orthogonality but do not guarantee a global minimum-distance solution.

2 Contributions

In our work, we propose a robust method that only assumes that the curve is three times differentiable. For a query point, we estimate a footpoint parameter by approximating a mapping function in which each planar position is assigned the parameter of the closest curve point. Figure 1a shows the signed distance field (SDF) for a continuous parametric curve. For comparison, Figure 1b shows the footpoint mapping for the same curve.

2.1 Footpoint mapping and its partial derivatives

Given a planar curve $C : [0, 1] \to \mathbb{R}^2$ and its footparameter mapping function $t : \mathbb{R}^2 \to \mathbb{R}$, we use the notation $C_i^{(n)}$ to denote the ith coordinate of $C^{(n)}(t(\mathbf{x}))$. Valasek et al. showed[5] that for any planar point $\mathbf{x}$, $\frac{\partial t}{\partial x_i}(\mathbf{x}) = C_i' H(\mathbf{x})^{-1}$, where $H(\mathbf{x}) = \|C'\|^2 - \langle \mathbf{x} - C, C'' \rangle$. Further differentiation gives $\frac{\partial^2 t}{\partial x_i \partial x_j}(\mathbf{x}) = (C_i'' C_j') H^{-2}(\mathbf{x}) - (C_i' C_j') \left(3 \langle C', C'' \rangle - \left\langle \mathbf{x} - C, C^{(3)} \right\rangle\right) H^{-3}(\mathbf{x})$. These results show that the footparameter mapping depends solely on the geometry of the curve and the distance between the query point and the curve. Once the mapping is known for a planar point, every partial derivative at that point can be computed using the curve geometry at the footparameter.

2.2 Our two-step method

Using the special structure of this footpoint mapping and several samples evaluated on the curve, we estimate the mapping as a polynomial function by Taylor expansion. This approach provides a strong initial candidate for iterative numerical solvers, ensuring that the initial parameter estimate is sufficiently close to the actual minimum-distance orthogonal parameter.

In the first step, we choose specific sample parameters on the curve. To capture the geometric features of the curve, we use a special parameter setting based on the following observations:

(a) We compute parameters corresponding to inflection points. As the curvature changes sign, the curve changes convexity, which cannot be accurately captured by second-order polynomial approximations. (b) The partial derivatives of t increase near the evolute of the curve, making the Taylor estimation numerically unstable and therefore less accurate. The evolute is closest to the curve (forming a cusp) when the curvature changes monotonicity. Both types of extrema can be computed numerically. If the curve is a rational polynomial, these sample-parameter estimations reduce to polynomial root finding. Empirically, we found that, to achieve reliable results, one extra sample should be inserted between the extrema samples. The first step serves as a curve preprocessing step and does not depend on any query point.

The second step estimates the mapping for each query point. Instead of simply estimating the Taylor polynomial from the sample points produced by the first step, we exploit the facts that (a) along the line in the direction of the curve's normal vector, this mapping function remains constant until we reach the cut locus, and (b) the closest point, which lies on that normal line, cannot be farther than the curvature radius.

The footparameter estimation for a query point is as follows. For each sample parameter, the normal line is estimated. This line is truncated to a half-line at the curvature radius. The query point is projected onto this half-line, yielding the closest point in the plane where the parameter equals the sample parameter (excluding possible cut-locus effects). At this line point, the derivatives of t are estimated, and the mapping parameter at the query point is estimated by a Taylor polynomial. After iterating over all sample parameters, the Taylor estimate that gives the best approximation (the curve point for the estimated parameter is the closest one) is selected. Finally, the approximation is fine-tuned with a variant of Newton iteration. In this second step, the computations for different query points are independent and use the output of the first step in a read-only manner. Therefore, the method is massively parallelizable.

3 Results

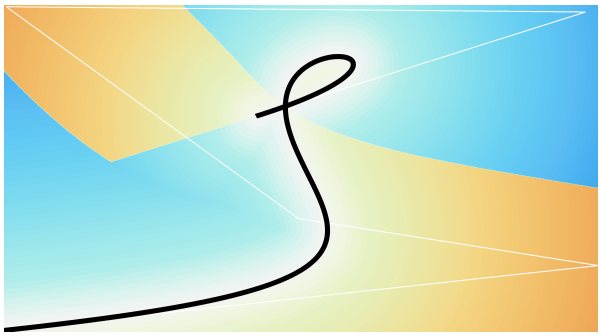
Figure 1a shows a quintic Bezier curve segment together with its control segments and the ground-truth signed distance field. This example represents a difficult case due to its self-intersection. In Figure 1b, the curve is colorized by its parameter, and the underlying footparameter mapping is visualized.

Figure 1c reports second-order footparameter mapping estimation and the samples chosen in the preprocessing step. Figure 1d presents the absolute error of the footparameter mapping estimation. Handling the cut locus is an inherent issue in SDF estimation: at such points, the closest-point mapping is not unique, the SDF is non-differentiable, and the footparameter mapping is discontinuous. The maximal error at a few query points near the cut locus can reach 0.9. However, the average error is 7×10^{-4}, with a standard deviation of 0.01. The median error is 5.8×10^{-5}. The figure clearly shows that the highest errors occur at the cut locus, especially at the cusp of the evolute. The error also increases at query points in the regions between sample points.

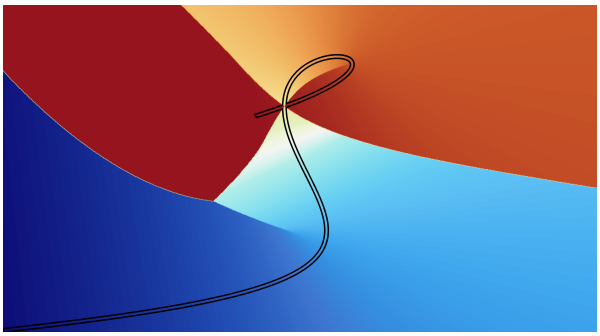
The final Signed Distance Field estimation is obtained using Newton iterations. We measured whether the Newton–Raphson iteration converges when starting from the estimated footparameter and how many iterations are needed to reach 10^{-7} accuracy. In most cases, the iteration count stayed below 5. However, for many query points near the cut locus, Newton–Raphson failed to converge.

In practice, it is common to use more advanced, higher-order Newton variants. We chose Kallay's second-order geometric Newton method[3]. We did not encounter any query point where the iterations diverged. The maximal iteration count was 15 (also close to the cut locus), but most cases required 0–3 iterations.

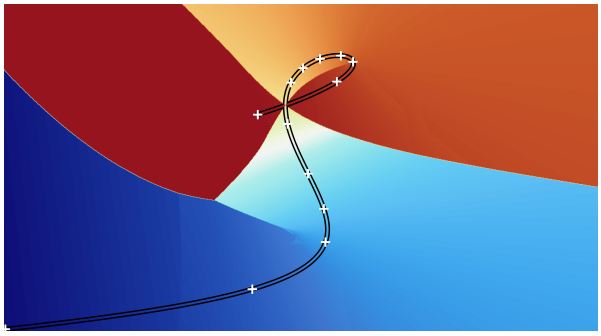
Finally, we answer the main question: how accurately can we estimate Signed Distance Fields? Figure 1g shows the final SDF estimation after geometric Newton refinement, and Figure



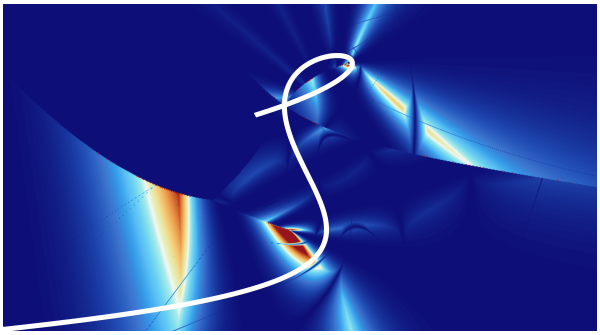
(a) Quintic Bezier curve, control segments, and its Signed Distance Field.



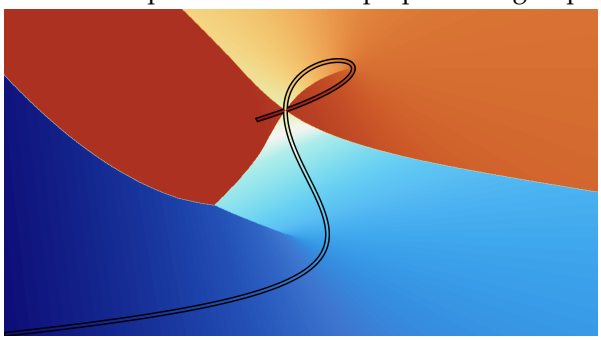
(b) Ground-truth footparameter mapping and curve parameterization.



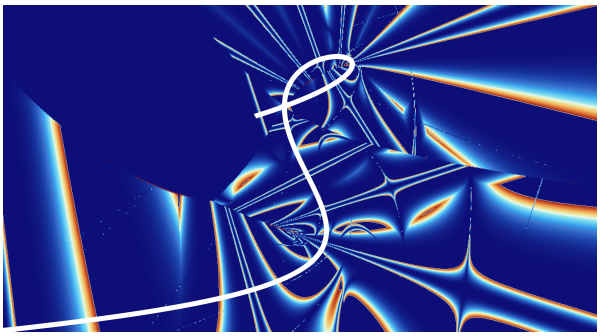
(c) Second-order footparameter mapping estimation and the samples chosen in the preprocessing step.



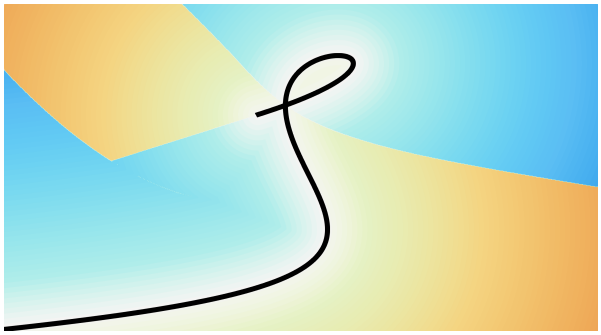
(d) Error of footparameter mapping estimation. Colormap scaled to $[0, 10^{-2}]$.



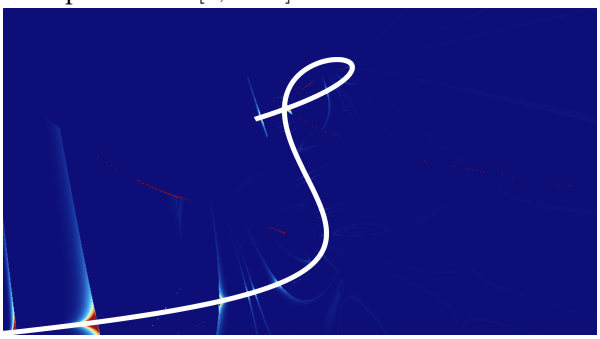
(e) Footparameter mapping estimation after geometric Newton refinement.



(f) Error of geometric Newton refinement. Colormap scaled to $[0, 10^{-6}]$.



(g) Signed Distance Field estimation after geometric Newton refinement.



(h) Error of SDF estimation after geometric Newton refinement. Colormap scaled to $[0, 10^{-9}]$.

Figure 1: Estimation of footparameter mapping.

1h shows its error. The average error was 5.23×10^{-5}, with a standard deviation of 6.26×10^{-3}. The median error was below 2×10^{-16}. All computations used 64-bit double-precision arithmetic.

4 Conclusion

We presented a two-step method for estimating point-curve minimal distance for general three-times differentiable planar curves. The method first samples the curve using its geometric features, then estimates the footparameter mapping by local Taylor approximations. The resulting parameter provides a reliable initial value for Newton-type refinement. Our experiments show that the method gives accurate SDF estimates, with the main errors concentrated near the cut locus, where the footparameter mapping is inherently discontinuous. Combined with geometric Newton iteration, the approach remains robust and requires only a few refinement steps for most query points.

References

[1] Xiao-Diao Chen, Jun-Hai Yong, Guozhao Wang, Jean-Claude Paul, and Gang Xu. Computing the minimum distance between a point and a nurbs curve. *Computer-Aided Design*, 40:1051–1054, 10 2008.

[2] Shi Min Hu and Johannes Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, 22:251–260, 3 2005.

[3] Michael Kallay. A geometric Newton–Raphson strategy. *Computer Aided Geometric Design*, 18(8):797–803, 10 2001.

[4] Xiaowu Li, Lin Wang, Zhinan Wu, Linke Hou, Juan Liang, and Qiaoyang Li. Convergence analysis on a second order algorithm for orthogonal projection onto curves. *Symmetry*, 9, 10 2017.

[5] Gábor Valasek, Csaba Bálint, and András Leitereg. Footvector representation of curves and surfaces. *Acta Cybernetica*, 25:555–573, 12 2021.

[6] Hongling Wang, Joseph Kearney, and Kendall Atkinson. *Robust and Efficient Computation of the Closest Point on a Spline Curve*. Nashboro Press, 2003.

Cross-Cultural Color Fidelity and Realism: A Comparative Perceptual Study across Real, Unreal Engine, and Unity Lighting Environments

Hanan Namrouti, Cecilia Sik-Lányi, Tibor Guzsvinecz

Abstract: Memory colours are canonical expectations for familiar objects and play a key role in colour constancy and visual preference. As colour-critical work increasingly moves into virtual environments, it is important to know whether real-time rendering engines preserve these memory-based judgements equally well for observers from different cultural backgrounds. This study compares colour perception in a physical 2700K viewing booth and matched simulations rendered in Unreal Engine and Unity. Sixty-two young adults ($N = 62$) from four origin regions (Europe, MENA, South Asia, Latin America) rated standard 24-patch ColorChecker targets and skin tones on multiple perceptual dimensions via structured questionnaire items. From these ratings we derived indices for memory colour rendition, global experience, appearance realism, contrast, skin rendering, and saturation. One-way and mixed-effects analyses showed robust effects of origin region on memory colour, skin rendering, and global experience, with South Asian participants generally giving lower scores than the other groups. Engine-related differences in saturation and contrast were largely shared across regions. These findings highlight that calibrated displays alone do not guarantee cross-cultural equivalence and motivate culturally adaptive colour pipelines in real-time rendering and virtual reality.

Keywords: memory colours, ColorChecker, virtual reality, cross-cultural perception, game engines

1 Introduction

Memory colours are long-term mental representations of the typical colours of familiar objects, such as bananas, blue sky, or human skin tones [1, 2, 4]. They function as perceptual priors that bias judgements toward canonical values and support colour constancy under changing illumination [3, 5, 6]. Apparent colour can therefore shift while familiarity remains stable, reflecting the influence of top-down expectations.

As colour-critical tasks migrate toward digital media, it is essential to understand how physical illumination compares with real-time rendering engines. Contemporary pipelines often rely on Unreal Engine or Unity, assuming that a single calibrated configuration is sufficient for a global user base [7, 8, 9]. However, cultural and geographical factors may modulate how observers evaluate appearance, realism, and skin-tone rendering, even when device calibration is held constant. This study compares colour perception across a physical 2700 K viewing booth and matched Unreal and Unity environments in a cross-cultural sample. We address two core research questions through formalized tracking:

- **RQ1**: Do high-level perceptual evaluations such as memory color perception, overall visual experience, and appearance realism differ systematically across participants from different global regions?
- **RQ2**: Does the effect of real-time rendering engine parameters on visual perception vary across cultural or regional groups, or do low-level visual attributes such as saturation and contrast influence observers uniformly regardless of origin?

To systematically investigate these questions, we establish two central hypotheses:

Hypothesis 1 (H_1): High-level perceptual evaluations, including memory color fidelity, appearance realism, and skin-tone naturalness, are significantly influenced by the observer's region of origin.

Hypothesis 2 (H_2): The perceptual effects of low-level rendering parameters — including saturation and contrast adjustments — interact significantly with cultural background, producing non-uniform sensory responses across regional groups.

2 Method

2.1 Participants and procedure

Sixty-two participants ($N = 62$, age 18–35 years) with normal or corrected-to-normal vision and normal colour vision (verified via Ishihara pseudo-isochromatic screening plates) took part. Based on self-reported nationality, observers were grouped into four distinct origin regions: Europe ($n = 20$), Middle East and North Africa (MENA, $n = 21$), South Asia ($n = 13$), and Latin America ($n = 8$). A within-subjects design compared three conditions: (1) **Real**, where a physical 24-patch standard ColorChecker chart was viewed in a controlled booth illuminated by a 2700K incandescent source; (2) **Unreal**, where the identical environment and ColorChecker target were modeled and rendered using Unreal Engine's ACES-based tone-mapping pipeline and displayed on a calibrated sRGB monitor simulating 2700K; and (3) **Unity**, where an equivalent digital scene was rendered in the Unity engine using its standard neutral tone-mapper under matching calibration. The stimulus set focused on the primary and secondary rows of the ColorChecker chart (red, green, blue, yellow–orange, gray patches) alongside multi-ethnic skin-tone configurations. For each environment, participants completed a comprehensive evaluation questionnaire, rating explicit behavioral items on 7-point Likert scales ranging from -3 (strongly negative/unnatural) to $+3$ (strongly positive/natural).

2.2 Measures and analysis

From the underlying raw items, we constructed six composite perceptual indices evaluated separately for each ambient condition. The specific constituent questions from the evaluation protocol were clustered as follows:

- *Memory Colour Rendition Index (MCRI)*: Aggregated from items checking *"[Color] patch looks familiar"* for the red, green, blue, and yellow ColorChecker samples, alongside direct appraisals of *"Natural familiar colors"*.
- *Global Experience*: Built from individual dimensional metrics evaluating environmental *"Brightness"*, *"Comfort"*, *"Quality"*, *"Engagement"*, and ambient *"Affect"*.
- *Appearance Realism*: Combining evaluations of whether *"Colors look..."*, *"Colors are..."*, spatial item tracking (*"Color edges are..."*), and structural rendering capabilities (*"Tell color apart"*).
- *Skin Index*: Clustered from dedicated biological surface targets tracking *"Lighter skin-tone natural"*, *"Darker skin-tone natural"*, and *"Skin-tone patches look..."*.
- *Saturation and Contrast Indices*: Extracted from combined sensory metrics detailing *"All colorful"*, isolated color band properties (*"[Reds/Greens/Blues/Orange] saturation"*), and *"Strong contrast"*.

We executed one-way ANOVAs on each index with Origin Region as the between-subject factor, followed by Bonferroni-corrected post-hoc comparisons. To resolve the systemic interactions, we fitted linear mixed-effects (LME) models setting Engine Condition and Origin Region as fixed factors.

3 Results

To evaluate the overarching impact of the investigated experimental factors across the primary psychophysical metrics, a series of linear mixed-effects models were executed.

Table 1 provides a comprehensive summary of these fixed-effect structures, degrees of freedom, F-values, and partial eta-squared (η_p^2) effect sizes, where individual subjects were modeled as baseline random intercepts ($\sigma^2_{\text{residual}} = 0.142$, $\tau_{00} = 0.058$).

Complementing these modeling parameters, Table 2 details the explicit mean $\pm$ SD perceptual index scores broken down across the four origin regions and three rendering conditions. Overall, these findings reveal systematic regional variations across several high-level cognitive measures—particularly for appearance realism, memory colour response, and skin-tone naturalness. In contrast, low-level sensory attributes, such as contrast and saturation, exhibit comparatively weaker and less consistent regional dependencies under the evaluated rendering pipelines.

Table 1: Summary of Psychophysical Indices and Linear Mixed-Effects Factor Evaluations with Effect Sizes (η_p^2).

Perceptual Index	**Fixed Factor**	**Num. df**	**Den. df**	F**-value**	p**-value**	η_p^2
Memory Colour Rendition Index (MCRI)	Condition	2	116	31.45	$< .001$	.35
	Region	3	58	7.89	$< .001$	.29
	Condition $\times$ Region	6	116	3.12	.006	.14
Skin Rendering Naturalness Index	Condition	2	116	42.18	$< .001$	.42
	Region	3	58	4.12	.009	.18
	Condition $\times$ Region	6	116	2.74	.014	.12

Table 2: Mean $\pm$ SD of perceptual index scores across origin regions under different conditions along with main and interaction effect sizes (η_p^2).

Index	**Condition**	**Europe**	**MENA**	**South Asia**	**Latin America**	**F**	**p**	η_p^2
Overall Experience Index	Real	1.40 ± 0.79	0.82 ± 1.17	0.31 ± 0.84	1.25 ± 0.94	3.75	.016	.12
	Unreal	1.11 ± 1.07	0.67 ± 0.99	-0.04 ± 0.87	1.28 ± 0.47	4.90	.004	.15
	Unity	0.89 ± 0.98	0.58 ± 1.08	-0.19 ± 0.92	0.88 ± 0.98	3.42	.023	.11
Appearance Realism Index	Real	0.20 ± 1.06	0.57 ± 1.09	-0.79 ± 1.17	0.71 ± 1.00	5.04	.004	.16
	Unreal	0.58 ± 0.95	0.44 ± 0.71	-0.69 ± 0.94	1.00 ± 0.64	8.92	$< .001$	.23
	Unity	0.83 ± 0.92	0.13 ± 1.33	-0.33 ± 1.36	1.00 ± 0.94	3.67	.017	.12
Contrast Index	Real	0.95 ± 1.23	0.84 ± 1.29	-0.13 ± 1.21	0.50 ± 0.93	2.39	.078	.08
	Unreal	1.42 ± 0.71	1.40 ± 0.84	0.33 ± 1.08	0.46 ± 1.57	5.16	.003	.17
	Unity	0.62 ± 0.77	0.78 ± 1.04	-0.13 ± 1.02	0.04 ± 1.98	2.25	.092	.07
MCRI	Real	0.82 ± 1.12	0.56 ± 0.93	-0.92 ± 1.41	1.53 ± 0.71	10.39	$< .001$	.29
	Unreal	0.89 ± 0.79	0.37 ± 0.85	-0.10 ± 0.70	0.38 ± 1.10	3.74	.016	.14
	Unity	1.43 ± 0.60	1.55 ± 1.13	0.04 ± 1.14	1.72 ± 0.99	8.17	$< .001$	.25
Skin Naturalness Index	Real	0.67 ± 0.87	0.46 ± 1.55	-0.46 ± 1.14	0.67 ± 1.15	2.56	.064	.09
	Unreal	0.98 ± 0.94	0.38 ± 1.06	-0.54 ± 0.87	0.75 ± 0.81	7.04	$< .001$	.21
	Unity	0.70 ± 1.42	1.21 ± 1.02	-0.72 ± 0.89	1.79 ± 0.69	11.01	$< .001$	.31
Saturation Index	Real	0.85 ± 0.88	0.43 ± 1.00	0.00 ± 1.09	0.88 ± 0.81	2.49	.070	.08
	Unreal	1.94 ± 0.56	1.67 ± 0.82	1.19 ± 0.92	2.00 ± 0.45	3.27	.027	.11
	Unity	0.71 ± 0.73	0.60 ± 0.94	-0.14 ± 0.93	0.71 ± 1.01	2.88	.044	.10

3.1 Regional effects on cognitive indices

Origin region significantly influenced high-level indices ($RQ1$). For MCRI, one-way ANOVAs showed robust regional differences across conditions [Real: $F(3, 58) = 10.39, p < 0.001$; Unreal: $F(3, 58) = 3.74, p = 0.016$; Unity: $F(3, 58) = 8.17, p < 0.001$]. Bonferroni post-hoc pairs verified that South Asian participants consistently assigned significantly lower MCRI scores than all other cohorts. Similar patterns emerged for Global Experience and Appearance Realism, where the South Asian cohort yielded noticeably negative bounds (e.g., -0.79 ± 1.17 for Real

Appearance), while Latin American participants recorded the highest overall affinity metrics. The Skin Index highlighted a crucial regional split, becoming highly pronounced in Unity [$F(3, 58) = 11.01, p < 0.001$], where Latin American (1.79 ± 0.69) and MENA (1.21 ± 1.02) cohorts rated digital skin reproduction with strong positive affect, whereas European (0.70 ± 1.42) and South Asian (-0.72 ± 0.89) cohorts found the rendering significantly less natural.

3.2 Engine effects on saturation and contrast

Addressing $RQ2$, low-level sensory attributes behaved in a highly uniform manner across demographics. The Saturation Index showed a dominant, highly significant main effect of Engine Condition [$F(2, 116) = 48.31, p < 0.001$], while the Condition $\times$ Region interaction was non-significant ($p = 0.42$). Every regional group experienced a substantial upward surge in perceived saturation within Unreal Engine (ranging from 1.19 to 2.00) compared to the physical baseline and Unity. The Contrast Index mirrored this exact pattern, peaking sharply in Unreal and dropping down across all regions in Unity.

3.3 Linear Mixed-Effects (LME) Modeling

To resolve the nested interaction paths of the repeated-measures configuration and quantify overarching parameter variances, linear mixed-effects models were executed. Table 1 summarizes the complete fixed-effect structures, degrees of freedom, and partial eta-squared (η_p^2) effect size values for the primary high-level psychophysical criteria, tracking subjects as baseline random intercepts ($\sigma^2_{\text{residual}} = 0.142, \tau_{00} = 0.058$).

For the *Memory Colour Rendition Index (MCRI)*, type III Wald F-tests presented in Table 1 indicate highly significant main effects for both Engine Condition ($F(2, 116) = 31.45, p < .001, \eta_p^2 = .35$) and Origin Region ($F(3, 58) = 7.89, p < .001, \eta_p^2 = .29$). Crucially, a significant **Condition $\times$ Region interaction** was confirmed ($F(6, 116) = 3.12, p = .006, \eta_p^2 = .14$), validating that cognitive adaptation gaps across different graphics engines depend systematically on the observer's regional background.

For the *Skin Rendering Naturalness Index*, the LME parameters detailed in Table 1 reveal a dominant main effect of Engine Condition ($F(2, 116) = 42.18, p < .001, \eta_p^2 = .42$) alongside a significant fixed effect of Origin Region ($F(3, 58) = 4.12, p = .009, \eta_p^2 = .18$). A significant **Condition $\times$ Region interaction** emerged ($F(6, 116) = 2.74, p = .014, \eta_p^2 = .12$), showing that regional expectations anchor the subjective acceptance thresholds for simulated biological surfaces across varying graphics pipelines.

4 Discussion and Hypothesis Evaluation

The findings provide clear evidence for distinguishing culturally mediated perceptual effects from uniform low-level rendering effects. Hypothesis 1 (H_1), which proposed that high-level perceptual evaluations—including memory colour fidelity, ColorChecker memory expectations, and skin-tone naturalness—are significantly influenced by the observer's region of origin, was fully supported. Significant regional main effects were observed for both the Memory Colour Response Index (MCRI) and Skin Naturalness Index ($p < .001$), and the linear mixed-effects (LME) modeling isolated significant Condition $\times$ Region interaction effects for MCRI ($p = .006$) and the Skin Index ($p = .014$), verifying that perceptual responses to individual rendering environments vary systematically across cultural groups.

These findings suggest that high-level colour judgments are shaped not only by physical display properties, but also by culturally informed cognitive expectations related to realism and colour fidelity. Conversely, Hypothesis 2 (H_2), which predicted that low-level rendering parameters like saturation and contrast would interact significantly with cultural background,

was not supported. Instead, strong main effects of rendering condition were observed uniformly across all participant groups, particularly for Saturation Index scores [$F(2, 116) = 48.31, p < .001$], while their respective interaction effects remained non-significant ($p = .42$). Unreal Engine consistently increased perceived saturation and contrast across all regional groups in a uniform manner, indicating that low-level rendering manipulations primarily affect baseline sensory appearance, whereas higher-level perceptual judgments remain culturally dependent. Overall, these findings suggest that globally deployed virtual and mixed reality systems should avoid assuming a universally optimal rendering configuration, and that implementing adaptive or region-sensitive rendering profiles may significantly improve perceptual realism and user experience across culturally diverse audiences.

References

[1] T. Hansen, M. Olkkonen, S. Walter, and K. R. Gegenfurtner, "Memory modulates color appearance," *Nature Neuroscience*, vol. 9, no. 11, pp. 1367–1368, 2006.

[2] M. Olkkonen, T. Hansen, and K. R. Gegenfurtner, "Color appearance of familiar objects: Effects of object shape, texture, and illumination changes," *Journal of Vision*, vol. 8, no. 5, article 13, pp. 1–16, 2008.

[3] D. H. Brainard and W. T. Freeman, "Bayesian color constancy," *Journal of the Optical Society of America A*, vol. 14, no. 7, pp. 1393–1411, 1997.

[4] C. Witzel, "Memory colors and color appearance," *Current Opinion in Behavioral Sciences*, vol. 30, pp. 1–6, 2019.

[5] D. Kersten, P. Mamassian, and A. Yuille, "Object perception as Bayesian inference," *Annual Review of Psychology*, vol. 55, pp. 271–304, 2004.

[6] M. D. Fairchild, *Color Appearance Models*, 3rd ed. Hoboken, NJ, USA: Wiley, 2013.

[7] M. Stokes, M. Anderson, and S. Chandrasekar, "A standard default color space for the Internet - sRGB," *IEEE Computer Graphics and Applications*, vol. 16, no. 1, pp. 25–31, 1996.

[8] International Commission on Illumination (CIE), *CIE 015: Colorimetry*, 4th ed. Vienna, Austria: CIE, 2018.

[9] E. Arabadzhiyska, A. Sitthi-Amorn, and M. Gross, "Perceptual color calibration for immersive displays," *ACM Transactions on Applied Perception*, vol. 14, no. 4, pp. 1–22, 2018.

Fully Parallel 3D Curve-Thinning on the FCC grid with Endpoint Re-Checking

Noel Nagy, Kálmán Palágyi

Abstract: Curve-thinning is an iterative object reduction aimed at extracting centerlines from binary objects in a topology-preserving way. In their previous work, the authors proposed the very first fully parallel 3D curve-thinning algorithm for $(18, 12)$ pictures of the face-centered cubic (FCC) grid. In this article, we adapt that existing algorithm to $(12, 18)$ pictures of the FCC grid. Both algorithms are based on asymmetric sufficient conditions for topology-preserving reductions that use the lexicographical order relation between two distinct points. In order to reduce the count of unwanted side branches in the produced centerlines, we propose alternating order relations that yield two additional algorithms. All four of our algorithms preserve the topology and guarantee one-voxel thin centerlines.

Keywords: Digital topology, Skeletonization, Curve-thinning, Face-centered cubic (FCC) grid

1 Introduction

Skeletonization plays a key role in a broad range of problems in image processing and computer vision [8]. It means extraction of *skeletal features* from (segmented) binary objects. The most important and most frequently used skeletal feature is the *centerline* that is a line-like 1D representation of both 2D and 3D objects [1].

Curve-thinning is an iterative object reduction for producing centerlines: the outermost layer of an object is deleted, and the entire process is repeated until stability is reached (i.e., no further points are deleted) [5, 6]. *Reduction* operators never change a *white* point in a *binary picture* to a *black* one, and a *parallel reduction* changes (or *deletes*) certain black points to white simultaneously [3]. A curve-thinning algorithm is said to be *fully parallel* if it applies the same parallel reduction at every iteration [3].

3D digital images are usually sampled on the regular *cubic grid* that is dual to $\mathbb{Z}^3$. In this work, however, our focus is on an unconventional 3D sampling scheme, the *face-centered cubic (FCC) grid* is taken into consideration [2, 6]. The point $(x, y, z) \in \mathbb{Z}^3$ is an element of the FCC grid if

$$(x + y + z) \mod 2 = 0\,.$$

On the FCC grid, the 12- and 18-adjacency relations are studied, see Fig. 1.

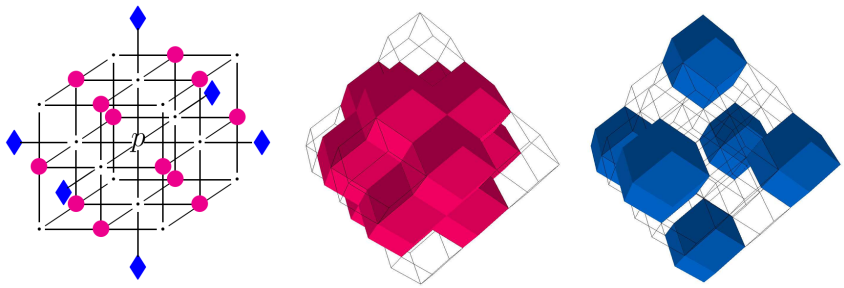


Figure 1: The twelve points marked '●' are 12-adjacent to the central point p, they and the six further points marked '♦' are 18-adjacent to p (left). (Note that points marked '·' are not in the FCC grid.) Each voxel (i.e., Voronoi cell) associated with these points is a rhombic dodecahedron (i.e., a convex polyhedron with 12 congruent rhombic faces).

In this work, we followed the seminal work of Kong and Rosenfeld on the fundamental concepts of digital topology [6], and $(m, n) \in \{(18, 12), (12, 18)\}$ pictures are considered in which

m-adjacency is assigned to the black points, and n-adjacency belongs to the white ones [2]. In this paper, we present four fully parallel 3D curve-thinning algorithms, acting on the FCC grid.

2 The Proposed Algorithms

All four of our algorithms follow the fully parallel thinning strategy, where each iteration step consists of two parallel reductions:

1. The first reduction deletes all black points that are not endpoints (i.e., they are not adjacent to exactly one black point) and satisfy the corresponding asymmetric point-based sufficient condition for topology-preserving parallel reductions [4]. Note that both conditions (those for the $(18, 12)$ and $(12, 18)$ pictures) also take into account the lexicographic ordering between two distinct points.

2. To reduce the number of unwanted side branches on the produced centerlines, the second reduction examines the endpoints generated by the previous reduction. In this endpoint re-checking phase, we delete all endpoints that have at least $k \geq 1$ neighbors deleted by the first reduction, where k is a pre-defined threshold. In our experiments, satisfactory results were obtained with $k = 1$.

Having outlined the common characteristics of our algorithms, we now present their specific features:

- The authors introduced Algorithm $FP(18, 12)$ in [7]. It is acting on $(18, 12)$ pictures of the FCC grid, and the first reduction of its iterations is based on Theorem 5.2 of [4].

- Algorithm $FP(18, 12)$-*alt* is derived from algorithm $FP(18, 12)$ by altering the direction of the lexicographic ordering relation in every second iteration step.

- Algorithm $FP(12, 18)$ is proposed for $(12, 18)$ pictures of the FCC grid, and it takes Theorem 5.4 of [4] into consideration.

- We derive algorithm $FP(12, 18)$-*alt* from algorithm $FP(12, 18)$ in the same way that algorithm $FP(18, 12)$-*alt* arose from algorithm $FP(18, 12)$.

We have managed to prove that all four proposed algorithms preserve the topology and produce one point thin centerlines for all possible pictures.

3 Results

In experiments our algorithms were tested on objects of different shapes. Here we present two illustrative examples, see Figs. 2 and 3. Each centerline was produced by choosing the endpoint re-checking parameter $k = 1$. The triplet (o, i, t) gives the count of object points, the required number of iteration steps, and the computation time (in msec) on a usual PC.

We can state that in all cases the algorithms $FP(18, 12)$, $FP(18, 12)$-*alt* are capable of producing well-positioned centerlines that contain all the essential branches and are free from unwanted side branches. It can also be stated that alternating lexicographic ordering resulted in significant improvement in the case of $(12, 18)$ pictures.

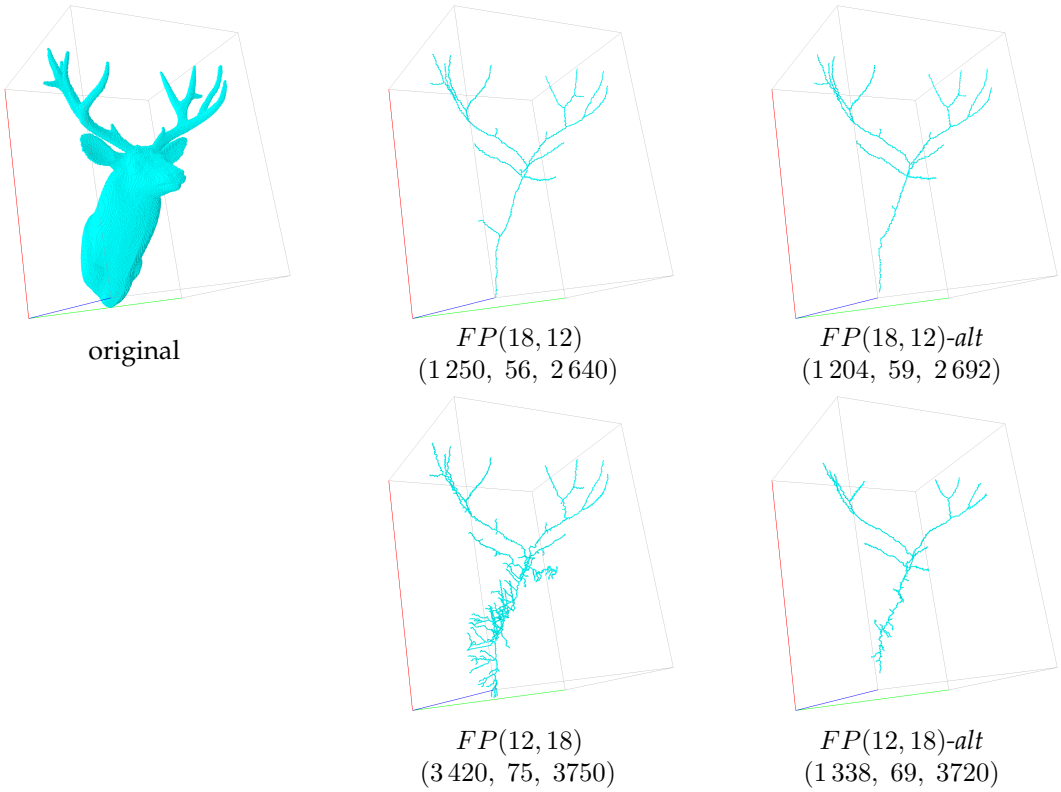


Figure 2: A 380×287×271 image of a deer and its centerlines produced by the four algorithms. The original image contains 904 144 object points.

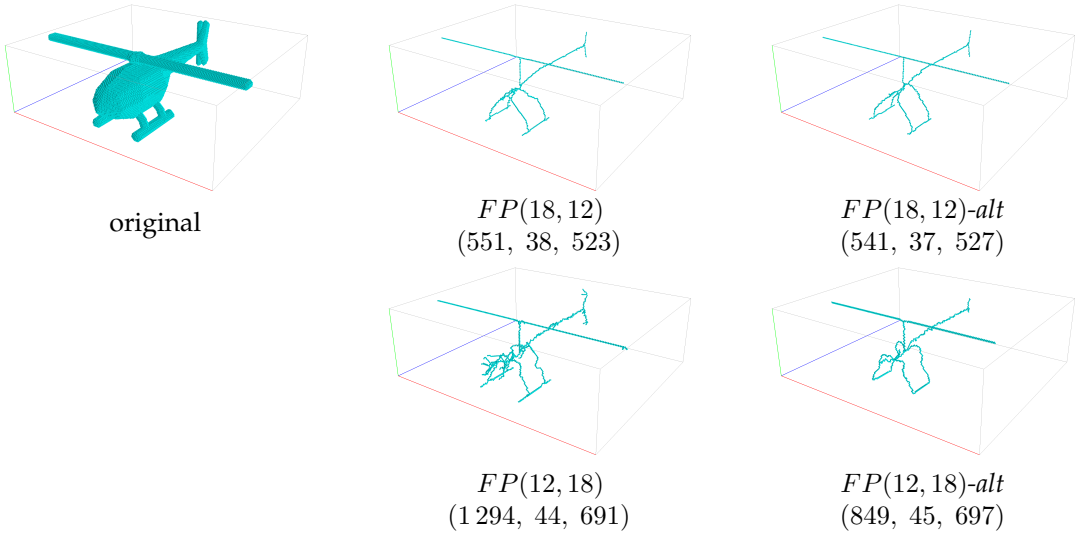


Figure 3: A 304×96×261 image of a helicopter and its centerlines produced by the four algorithms. The original image contains 189 077 object points.

References

[1] Cornea, N., Silver, D., Min, P.: Curve-skeleton properties, applications, and algorithms. IEEE Transactions on Visualization and Computer Graphics **13**, 530–548 (2007). doi:10.1109/TVCG.2007.1002

[2] Gau, C.J., Kong, T.Y.: Minimal nonsimple sets of voxels in binary images on a face-centered cubic grid. International Journal of Pattern Recognition and Artificial Intelligence **13**, 485–502 (1999). doi:10.1142/S021800149900029X

[3] Hall, R.W.: Parallel connectivity-preserving thinning algorithms. In: Kong, T.Y., Rosenfeld, A. (eds.) Topological algorithms for digital image processing, pp. 145–179. Elsevier Science, Amsterdam (1996). doi:10.1016/S0923-0459(96)80014-0

[4] Karai, G., Kardos, P., Palágyi, K.: Sufficient conditions for topology-preserving parallel reductions on the face-centered cubic grid. Journal of Mathematical Imaging and Vision **66**, 271–292 (2024). doi:10.1007/s10851-024-01177-y

[5] Kong, T.Y.: On topology preservation in 2-d and 3-d thinning. International Journal of Pattern Recognition and Artificial Intelligence **9**, 813–844 (1995). doi:10.1142/S0218001495000341

[6] Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing **48**, 357–393 (1989). doi:10.1016/0734-189X(89)90147-3

[7] Nagy, N., Palágyi, K.: Fully Parallel 3D Curve-Thinning on (18,12) Pictures of the FCC Grid In: Balázs, P. et al. (eds.) Combinatorial Image Analysis – 23rd International Workshop, IWCIA 2025, LNCS **15985**, Springer (2026). in press. doi:10.1007/978-3-032-19347-6_5

[8] Saha, P.K., Borgefors, G., Sanniti di Baja, G.: Skeletonization: Theory, methods and applications. Academic Press, London (2017). doi:10.1016/B978-0-08-101291-8.00017-1

Athlete Tracking and Identification in Water Polo

Gergő Papp, Emil Világos, Krisztián Szekeres, Ellák Somfai

Abstract: Athlete tracking and re-identification are fundamental to extract key insights from sports videos, such as estimating the total distance covered by players or analysing team tactics. However, the aquatic environment of water polo introduces unique computer vision challenges that render standard land-based tracking algorithms insufficient. These include frequent occlusions due to partial or total submersion, visual noise from water splashes, and image degradation caused by reflections and glare. Furthermore, the non-linear movement dynamics of athletes in water differ significantly from terrestrial sports, complicating trajectory prediction.

Due to the absence of publicly available annotated datasets for water polo, this study utilizes a curated and annotated dataset specifically designed for aquatic sports analysis. We propose a baseline pipeline for player detection, tracking, and team classification. Our methodology achieves tracking performance with a ***MOTA*** (Multiple Object Tracking Accuracy) of 0.642 and an ***IDF1*** of 0.606, alongside near-perfect accuracy in team and role identification. These results can further be used to gain insights of the game dynamics and athlete statistics. This work establishes a benchmark for water polo analytics and highlights the aquatic domain as a significant frontier for future computer vision research.

Keywords: Computer Vision, Deep Learning, Multi-Object Tracking, Deep Metric Learning, Multi-Task Learning, Re-Identification, Team Affiliation, Water Polo, Sport Analysis

1 Introduction

In recent years, sports organizations and teams have shown increasing interest in collecting athlete-centric data to gain deeper insights into performance and game dynamics. Such analytics support a wide range of applications, including team coaching, talent scouting, and enhanced fan engagement through personalized content. Wearable sensor-based tracking systems can provide accurate measurements, but they require athletes to wear dedicated equipment. These systems can work well in most land sports, but water degrades or absorbs GPS and other radio signals.

Optical tracking systems offer an alternative. A fundamental component of such systems is object tracking , which aims to continuously identify players and other relevant objects throughout a video. Tracking provides a spatio-temporal representation of the game, which can reveal tactical patterns, and also allows the extraction of personalized statistics. Reliable tracking is challenging in sports environments, because players have non-linear motion, frequently occlude each other, and have very similar intra-class visual appearances. To address this problem, re-identification techniques are commonly used to match the same player across different frames or camera views. Assigning each detected player to team A or B can improve the robustness of tracking and player identification. This task is complicated because of variations in player pose, occlusions, and uneven illumination commonly present in sports footage. Team membership classification is also essential for many downstream analyses, such as analysing team formations and tactics, or for gathering team level statistics. In practice, an effective sports tracking system must jointly address multi-object tracking, player re-identification, team affiliation, and role classification.

An additional challenge in distinguishing players in water polo videos is the lack of stable and clearly recognizable visual features on their bodies. In practice, only the caps carry relevant information, but their size is small, and the numbers on them are often blurred, occluded or difficult to read. Due to the above, most of the tracking, re-identification and team classification

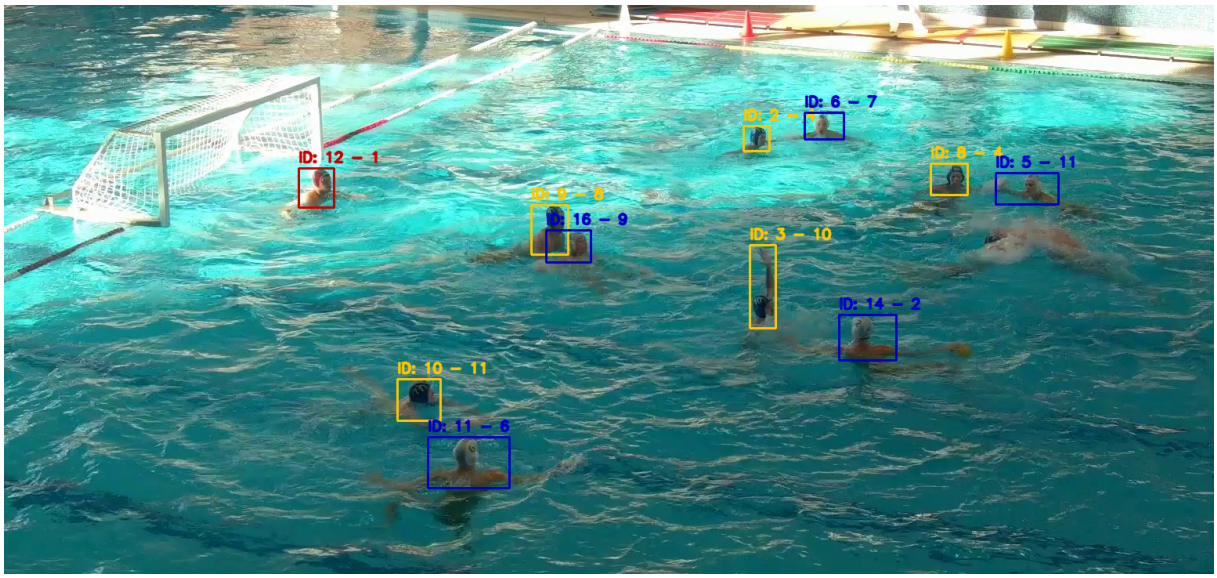


Figure 1: Detected and tracked players: red indicates the goalkeeper, while blue and yellow represent teams A and B. Each track is labeled with a unique track ID and the detected player number.

models that are successfully used in the analytics of other sports do not work reliably in this environment.

To the best of our knowledge, this work introduces the first machine learning pipeline for water polo video analysis, combining player detection, team affiliation, number recognition, role classification (Player/Goalkeeper), and athlete tracking within a single framework.

2 Proposed Methods

Athlete Detection: We employ RT-DETRv2 [2], fine-tuned on our custom water polo dataset, to detect players on the water surface. We trained the model to simultaneously detect full-body bounding boxes for the parts visible above the water, as well as separate head bounding boxes for each player. Head detections are particularly important, as they provide the only reliable visual information for determining both player role and team affiliation, which are then used for re-identification. We also experimented with using full-body bounding boxes for team affiliation, role classification, and re-identification. However, global embeddings extracted from full-body crops proved less robust, as they are often affected by player overlap or the presence of other objects within the bounding box. In contrast, focusing only on head regions simplifies the problem, making head detections essential for accurate and stable player representation. To associate head detections with full-body bounding boxes, we use the Intersection over Smaller Area (IoS) metric combined with Hungarian matching.

Team Affiliation and Role Classification: We trained a custom convolutional neural network using a multi-task learning for team affiliation and role classification. The proposed architecture consists of a shared feature extractor backbone and two task-specific heads: one dedicated to metric learning and the other to role classification. The backbone takes as input a cropped image of a player's head and outputs a compact feature embedding, which is jointly optimized and utilized by both downstream tasks.

The backbone is composed of three convolutional layers with 16, 32, and 64 channels, respectively, each using 3×3 kernels. Each convolutional layer is followed by a batch normalization layer, a non-linear activation function and a pooling layer to progressively reduce the spatial resolution while increasing the level of abstraction. The entire network is trained end-to-end

using the AdamW optimizer, with a combined loss function that balances the objectives of both tasks.

For the metric learning branch, we employ the Batch Hard Triplet Loss to learn an embedding space where players from one team are closer together, while players from different teams are pushed apart. During training, each mini-batch is constructed by randomly sampling P teams and selecting K samples per team. From these samples, triplets are formed consisting of an anchor, a positive example (from the same team), and a negative example (from a different team). The batch hard strategy selects the most challenging examples within the batch. To form the positive pair, it selects the sample (from the same team) which is farthest from the anchor in the embedding space, while for the negative pair it selects the sample (from any different team) that is closest to the anchor. This encourages the model to learn more robust and discriminative features. The loss function minimizes the distance between anchor-positive pairs while maximizing the distance between anchor-negative pairs.

The role classification head performs binary classification into two classes: player and goalkeeper. Due to class imbalance goalkeepers appear significantly less frequently than field players, so we use focal loss to emphasize harder and underrepresented examples during training. This auxiliary task also enriches the feature representations by incorporating role-specific semantic information into the shared embedding space.

Because both tasks rely heavily on colour information, using colour-based data augmentation during training was essential to improve generalization. Specifically, we applied a range of colour transformations, including adjustments to hue, brightness, contrast, saturation, and colour temperature. These augmentations help the model become more robust to changes in lighting conditions, camera settings, and visual appearance differences across matches.

Tracking: For tracking, we utilize DiffMOT [1], a real-time diffusion-based multi-object tracker, to tackle the complex non-linear motion patterns of players. The original tracker relies on deep features for associating existing tracks with new detections. However, in this domain, intra-class embeddings are often highly similar, making association less reliable.

To address this limitation, we modify the association by incorporating additional semantic information, including team affiliation, player number and role. Player numbers are obtained using CLIP4STR [3], which is applied earlier in the pipeline.

Furthermore, we add a voting-based filtering mechanism into the tracking process, which helps to suppress noise and smooth out team, role, and player number data.

3 Experimental Results

We created a custom dataset of 10 video sequences ($\approx 17\,000$ frames), as no annotated data are publicly available for water polo. The dataset was split into 6 training sequences and 4 validation sequences. Due to the limited dataset size, we do not have a separate test set. All reported metrics in Table 1 and Table 2 are computed per video and averaged across the validation sequences, providing an estimate of the pipeline's ability to generalize beyond the training data. Because there is no independent test set, some hyperparameters may be slightly biased toward the validation set.

The tracking results show that the pipeline can reliably maintain player identities across frames, even when detections are imperfect. However, due to the presence of water, it is difficult to define precise object boundaries, as parts of the players are often visible beneath the surface. As a result, bounding box annotations may vary, since clear edges are not well defined. This makes it challenging for the detector to consistently determine where objects end, particularly since the goal is to detect only the portions above the water. This issue also impacts the evaluation metrics.

Raw Model Outputs		
Model	**Metric**	**Value**
RT-DETRv2	mAP@0.5	0.702
Custom Model	Role Accuracy	0.994
	Team Accuracy	0.848
CLIP4STR	Number Accuracy	0.391

Table 1: Performance results of used models on the validation set. Due to the ambiguity of defining the player bounding boxes in water polo, for detection benchmark the mean average precision was evaluated only at IoU=0.5.

Pipeline Outputs		
Component	**Metric**	**Value**
Detection	mAP@0.5	0.729
Tracking	MOTA	0.642
	IDF1	0.606
Recognition	Role Accuracy	0.997
	Team Accuracy	0.899
	Number Accuracy	0.592

Table 2: Performance results of the proposed pipeline on the validation set including filtering techniques.

Role and team recognition achieve very high accuracy on our validation set. It is important to note that the metrics in Table 2 are computed on the outputs of the entire pipeline, including filtering techniques that help to improve consistency. Number recognition is more challenging, due to small player numbers on caps, motion blur, and occlusions, but the integrated approach of the pipeline improved significantly this metric as well.

Overall, the results demonstrate that the proposed pipeline is capable of detecting, tracking, and recognizing player roles, teams, and numbers in water polo videos.

Acknowledgements

This work was supported by the University Research Scholarship Program, funded by the Ministry of Culture and Innovation of Hungary and the National Research, Development and Innovation Fund. The authors also wish to acknowledge OX-IT, the owner of the project, for their support throughout this work.

References

[1] W. Lv, Y. Huang, N. Zhang, R.-S. Lin, M. Han, and D. Zeng, "Diffmot: A real-time diffusion-based multiple object tracker with non-linear prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 19321–19330.

[2] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer," *arXiv preprint arXiv:2407.17140*, 2024.

[3] S. Zhao, R. Quan, L. Zhu, and Y. Yang, "Clip4str: A simple baseline for scene text recognition with pre-trained vision-language model," *IEEE Transactions on Image Processing*, vol. 33, pp. 6893–6904, 2024.

Spline Proxies for Planar Level Set Distance Computations

Anna Lili Horváth, Csongor Csanád Karikó, and Gábor Valasek

Abstract: We propose a spline-approximation framework for computing signed distances to algebraic implicit shapes in the plane. Within a user-specified region of interest, we sample the boundary by casting uniform random rays and collect intersection points. We establish the topology between the samples with respect to the boundary with a numerical tracing method, while marking potential self-intersections. This allows us to fit parametric polynomial curves and use them in lieu of the implicitly defined boundary in closest point queries. This reduces the problem of finding closest points on implicit algebraic boundaries to a univariate polynomial root finding problem.

1 Introduction and Related Work

We consider the continuous distance transform of implicitly defined shapes in the plane, in particular, we compute the signed distance to boundaries defined by $f(x, y) = 0$. Traditional computational methods approach this in the discrete setting by solving a distance propagation model at a finite set of samples. Continuous approaches compute distances via closest-point constructions on implicitly defined boundaries [1]. Our goal is to establish a continuous apparatus that computes the closest points on the boundary via parametric curves.

Our main contribution is an approximate conversion from implicit boundaries to parametric curves, preceded by a probabilistic sampling of topological components [2]. In particular, we present the following framework: i) topological coverage is facilitated via statistical means, namely that every boundary is captured with probability proportional to its arc length, and ii) the componentwise boundaries are captured faithfully via a piecewise parametric curve approximation. In runtime, the spline approximants are used to compute distances, for which robust solutions exist.

This transforms the notoriously difficult problem of finding closest points on an algebraic polynomials – a two-variate constrained optimization problem – to univariate root finding. This follows from that distance computations to polynomial curves reduces to finding the roots of a univariate polynomial, for which robust, high performance solutions exist [3].

2 Spline-Based Isocontour Distance Computation

The input shape is defined by $\{f = 0\} = \{\mathbf{x} \in \mathbb{R}^2 \mid f(\mathbf{x}) = 0\}$. We assume that f exposes an $f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}$ value and $\nabla f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^2$ gradient evaluation interface, and means to compute all intersections of $\{f = 0\}$ with a parametric ray $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}$, where $\mathbf{p}_0 \in \mathbb{R}^2$ is the origin of the ray and $\mathbf{v}$, $|\mathbf{v}| = 1$ is its direction vector. We assume that the user supplied a region of interest (ROI) via an axis-aligned bounding box $\mathbf{b}_{min}, \mathbf{b}_{max} \in \mathbb{R}^2$. For the remainder, we assume $f(x, y)$ to be algebraic, i.e., $f(x, y) = \sum_{i,j} a_{ij} x^i y^j$, however, not necessarily represented by its a_{ij} monomial coefficients.

We compute the SDF of $\{f = 0\}$ by extracting a piecewise polynomial approximation to the boundary in a preprocessing step. During runtime, we compute distances to these parametric boundaries to approximate the distance from $\{f = 0\}$.

2.1 Boundary Sampling

First, we establish an initial sampling of the boundary. Since $\{f = 0\}$ may consist of multiple disjoint parts, we need a method that is capable of capturing points on every component boundary. We chose Ling et al.'s [2] algorithm that provides a uniform random sampling of the

boundary by computing surface-line intersections from uniformly distributed lines in the ROI. Here, the expected aggregated intersection points sample connected components in proportion to their arc lengths.

We expose the number of rays as a parameter to the user. We register every intersection of the uniform random rays with $\{f = 0\}$. Intersections are computed by solving the polynomial $f \circ \boldsymbol{p}(t) = 0$ with Yuksel's method [3]. The polynomial is obtained by fitting $n + 1$ samples along the ray, taken at Chebyshev node parameters, where n is the total degree of the particular algebraic shape.

2.2 Boundary Exploration

2.2.1 Dense Boundary Component Sampling

The previous stage generated uniform random samples on the intersected boundary components. This step makes these samples approximately uniform by distance per component, by taking $\epsilon > 0$ length steps in the $\boldsymbol{t}_i$ tangent direction and projecting the result back to the boundary to obtain new samples. The exploration steps when we reach an existing sample, which we denote by $r(\boldsymbol{p}_i, \boldsymbol{t}_i)$ for $\boldsymbol{p}_i$ starting point.

If we explore the boundary in both the $\boldsymbol{t}_i$ and $-\boldsymbol{t}_i$ directions from each sample, we get an adjustable density point set on the component boundaries. Here, $\epsilon > 0$ is a user parameter, that can be tuned to the particular $f(x, y)$ input function manually.

2.2.2 Connectivity

The algorithm used in boundary component sampling also allows us to establish connectivity between the dense samples. A dense sample $\boldsymbol{p}_i$ is connected to dense samples $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$ if $\boldsymbol{r}(\boldsymbol{p}_i, \boldsymbol{t}_i) = \boldsymbol{p}_j$ and $\boldsymbol{r}(\boldsymbol{p}_i, -\boldsymbol{t}_i) = \boldsymbol{p}_k$. If we reach the bounding box in one of these directions, the point has one less neighbor.

At self-intersections, the path of the exploration and the resulting neighbors might not be the complete topologically. Consequently, the connectivity relation is not symmetric. We rely on this to mark singularities, as shown next.

2.2.3 Detecting Self-Intersections

Singularities that make high-quality isocontour approximation difficult come in many forms and typically require case-specific, higher-order analysis of $f(x, y)$. For instance, $\nabla f = \mathbf{0}$ is a necessary condition for self-intersections, while differential geometric vertices correspond to curvature extrema and discontinuities. Here we focus on self-intersections only. At these singularities, boundary exploration can yield inconsistent neighbor relations. We exploit these inconsistencies to detect self-intersections.

A dense sample $\boldsymbol{p}_i$ is marked as a potential singularity if it has inconsistent neighborhood, that is, $\exists \boldsymbol{p}_j : \boldsymbol{r}(\boldsymbol{p}_j, \pm\boldsymbol{t}_j) = \boldsymbol{p}_i \wedge \boldsymbol{r}(\boldsymbol{p}_i, \pm\boldsymbol{t}_i) \neq \boldsymbol{p}_j$. The marked points are not necessarily at the exact self-intersections and multiple neighboring points may get labeled as such at high density sampling. In the following step, we reduce the number of samples and only keep one such point for each intersection.

2.2.4 Coarsening Boundary Samples

We establish a new sampling distance, $\epsilon' > 0$, and remove samples until the distance between any two neighboring samples is at least ϵ', $\epsilon' = \frac{3}{2}\epsilon$ used as a default. We implement this phase by repeatedly deleting points closer than ϵ' while keeping potential singularities, then recalculating neighbours and singularities. We repeat the algorithm until convergence.

This enforces ϵ'-separation and leaves at most one marked sample per ϵ'-neighborhood. Thus, ϵ' must be smaller than the distance between distinct singularities to avoid merging them.

2.3 Spline Approximation of the Boundary

2.3.1 Selecting Fitting Sample Pairs

Given the sampled points, neighbor connectivity, and detected singularities, we next choose samples to connect with parametric curve segments. To preserve the discrete topology, segment endpoints are constrained to the original samples. Since neighborhoods are unreliable near singularities, we exclude all such marked samples as segment progenitors. We fit segments between unmarked neighbouring points and between unmarked points and a neighbouring singularity, this ensures that all branches are covered at singularities.

This fits each boundary section exactly once and covers all sections incident to singularities. Cases where two adjacent samples are both marked singular are not handled. In practice, we avoided this by choosing a sufficiently small step size.

2.3.2 Fitting a Parametric Segment

For a neighbor pair $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ selected for fitting, we consider the line segment between $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ and evaluate it at fixed parameters, using the boundary exploration mentioned previously. Then we use a modified LSQ method to approximate the boundary points with a curve segment. The LSQ method guarantees the interpolation of the start and end points between connected segments.

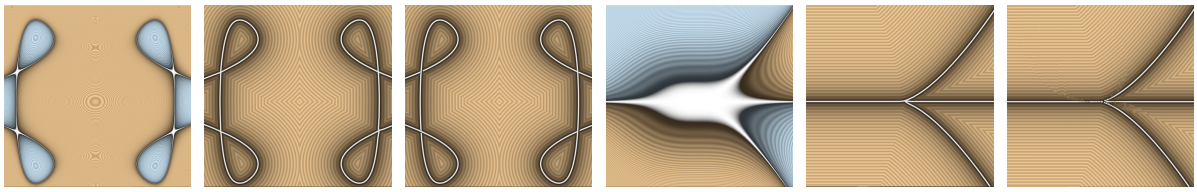
2.4 Distance Computation

Finding the closest point of a $\boldsymbol{b}(t)$ curve to a $\boldsymbol{x}$ query point can be formulated as a minimization problem: $\arg\min_{t\in[0,1]} \|\boldsymbol{b}(t) - \boldsymbol{x}\|^2$. Since its derivative, $2(\boldsymbol{b}(t) - \boldsymbol{x})^T \cdot \boldsymbol{b}'(t)$ is a degree $2n-1$ polynomial, we can compute its roots with Yuksel's method. We evaluate the curve at each root and take the minimum distance from the computed points and the end points. We accelerate this by adding the axis-aligned bounding boxes of the curves to an R-tree. Then first we find the closest AABB to the query point $\boldsymbol{x}$ and compute the d minimum distance from its curve. Next, we query all AABBs which intersect the $[x_x - d, x_y - d, x_x + d, x_y + d]$ box. In the second stage, we iterate over all such AABBs, first computing the d_{AABB} distance of the current AABB to $\boldsymbol{x}$ and only computing the curve-point distance if d_{AABB} is less then our current lowest distance during the minimum search. Sign can be taken from $f(\boldsymbol{x})$.

3 Test Results

We tested our method using a CPU implementation, by computing distances on an 1920×1080 grid, using single precision floating point arithmetic for all methods. We compared our method with Saye's [1] Newton implementation, referred to as Saye's Newton. Both methods were initialized with intersections from 512 random rays. We fit quintic Bézier curves to 8 samples per proxy.

For validation, we evaluated both methods on circle scenes with known ground-truth distances: a single circle, and the union and intersection of two circles. For each distance field, we compared against the exact distance at every sample and report maximum, median, and RMS errors in the table below.



(a) Barth's sextic (b) Our method (c) Saye's Newton (d) Harlekin (e) Our method (f) Saye's Newton

Figure 1: Real-polynomial tests. Left: the implicit polynomial. Our method casts 32 random rays to generate 100 boundary samples, then fits parametric segments. Saye's Newton method uses a denser random sample set of 512 rays. Pixels are colored with function (first column) and SDF values (second and third column). Our method is more accurate and remains robust where Saye's Newton fails, e.g., the self-intersection in Barth's sextic and the junction in the Harlekin.

	Circle		Union of circles		Intersection of circles	
	RMS	max	RMS	max	RMS	max
Our method	1.19×10^{-5}	7.90×10^{-5}	1.23×10^{-4}	1.31×10^{-3}	1.20×10^{-4}	6.93×10^{-4}
Saye's Newton	2.04×10^{-6}	7.21×10^{-6}	1.82×10^{-3}	3.40×10^{-2}	3.11×10^{-3}	2.87×10^{-2}

Both methods depend on sampling density, and Saye's Newton method may require many samples for robustness. We chose visually reasonable sample counts and step sizes, noting that the two methods use different sample sets. On simple scenes, our accuracy matches Saye's Newton method. On shapes with corners and sharp features, we reduce mean and maximum error with comparable median error.

After the validation, we ran our method for complex algebraic scenes. Figure 1 shows the test shapes and distance fields calculated with the different methods. These examples show that our method is able to handle details that the Saye's Newton method cannot, even with a large sample number. Finally, we profiled our method, see the following table. Boundary exploration is dominated by topology reconstruction ($\approx 50\%$) and point reduction ($\approx 40\%$).

	Our method				Saye's Newton	
Scene	Boundary sampling	Boundary exploration	Fitting	Distance query	Boundary sampling	Distance query
Barth's sextic	0.2321	5.7838	0.8763	0.0016	3.2145	0.00015
Harlekin	1.1697	1.7474	0.5171	0.0014	1.6728	0.00016

4 Conclusions

We presented a method to approximate signed distances to planar algebraic implicit curves by fitting parametric segments to a feature-preserving sample set and evaluating distances to these splines. On complex scenes, this reduces error and captures features where Saye's multivariate Newton-based distance evaluation fails such as self-intersections. Future work includes a GPU implementation, automatic step-size selection for curve sampling, adaptive per-segment spline degrees, and improved singularity handling.

References

[1] Saye, Robert I. High-order methods for computing distances to implicitly defined surfaces. Communications in Applied Mathematics and Computational Science. 2014

[2] Ling, Selena and Madan, Abhishek and Sharp, Nicholas and Jacobson, Alec. Uniform Sampling of Surfaces by Casting Rays. Computer Graphics Forum. 2025

[3] Yuksel, Cem. High-Performance Polynomial Root Finding for Graphics. Proc. ACM Comput. Graph. Interact. Tech. 2022

Fuzzy Rule-Based Smart Street Lighting for Energy-Efficient Urban Illumination

Ádám Brand, Attila Magyar

Abstract: This paper presents a fuzzy rule-based smart street lighting system designed to minimize the energy consumption of LED street lamps under varying weather conditions while maintaining adequate visibility. The controller employs fuzzy soft computing subsystems to determine weather conditions from sensor data, estimate visibility, and set the forward current of lamps accordingly. The system architecture consists of two interconnected systems: one with sensors and environmental evaluators and one controlling the lamp current. Simulation studies in MATLAB/Simulink under three scenarios – clear summer night, summer night with thunderstorm, and foggy night – demonstrate that approximately 20% energy savings can be achieved under normal conditions. The approach is cost-effective, adaptive, and robust to environmental uncertainties, making it suitable for smart city applications.

Keywords: fuzzy logic control, adaptive lighting, energy efficiency, MATLAB/Simulink simulation

1 Introduction

The concept of smart cities focuses on improving sustainability, energy efficiency and quality of life in urban environments. Street lighting represents a significant portion of a city's electricity consumption and can account for up to 60% of nighttime energy usage [1]. Consequently, improving the efficiency of public lighting systems is an important step toward sustainable urban infrastructure.

Traditional street lighting systems typically operate using time-based schedules or twilight sensors. While these methods are simple and reliable, they lack adaptability to environmental conditions such as fog, rain, or sudden weather changes. As a result, energy may be wasted during periods when full illumination is unnecessary.

Recent research has explored intelligent control approaches for street lighting. Adaptive systems based on wireless sensor networks and LED technology have been proposed to improve efficiency and flexibility [2]. Traffic-aware lighting control has also been investigated using distributed sensors to dynamically adjust illumination levels [3]. Other approaches employ artificial intelligence techniques, such as neural networks or fuzzy logic, to determine optimal lighting levels [4].

Among these methods, fuzzy logic provides a particularly attractive solution because it can handle uncertainty and imprecise sensor information effectively. Furthermore, fuzzy systems can incorporate expert knowledge in the form of linguistic rules, making them well suited for environmental control applications.

In this paper, a fuzzy rule-based smart street lighting system is proposed. The system evaluates weather conditions and visibility using sensor measurements and adjusts the forward current of LED lamps accordingly. Simulation studies demonstrate that the proposed approach can significantly reduce energy consumption while maintaining acceptable illumination levels.

2 Proposed Smart Street Lighting System

The system divides into two connected subsystems (see Figure 1):

SYSTEM A contains the LED lamp with controllable forward current, sensors, and fuzzy soft computing modules: the Weather Condition Evaluator (WCE) and the Visibility Condition Evaluator (VCE).

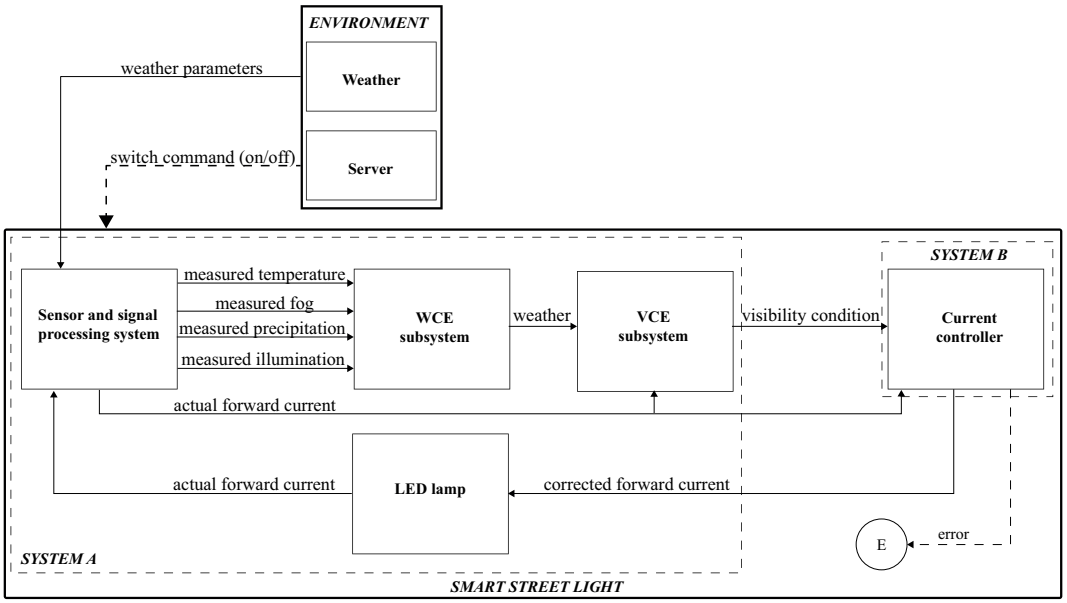


Figure 1: Structure of the proposed smart street light system [5]

SYSTEM B contains the Current Controller, which determines the lamp's forward current based on input from SYSTEM A.

2.1 Sensors and Adjustable Parameters

The LED lamps are equipped with:

- Illuminance sensor (0–100 relative scale)
- Temperature sensor (-30°C to +60°C)
- Optical rain sensor (0–100 relative scale)
- Fog sensor (0–100 relative scale)

The main adjustable parameter is the lamp forward current. Correlated color temperature is not considered in this study.

2.2 Fuzzy Soft Computing Subsystems

Each subsystem employs Mamdani-type fuzzy inference with centroid defuzzification:

- **WCE:** determines the current weather state from sensor inputs.
- **VCE:** evaluates visibility based on weather, lamp state, and time.
- **Current Controller:** sets the forward current of the lamp to maintain optimal visibility while minimizing energy usage.

Membership functions are triangular, trapezoidal, S- or Z-shaped, chosen for efficient control. Rule bases are designed to respond to common weather conditions (clear, cloudy, rainy, snowy, foggy).

3 Simulation Case Studies

A MATLAB/Simulink model evaluated the system performance under three scenarios with *simulated sensor data according to the following weather conditions*:

Scenario 1: Clear summer night The system maintains optimal visibility with minimal forward current. Energy savings are maximized when weather is stable.

Scenario 2: Summer night with thunderstorm The system responds to increased precipitation and reduced illuminance by adjusting forward current appropriately.

Scenario 3: Foggy night High fog reduces visibility; the controller increases current but may be limited by maximum available lamp current, resulting in reduced energy savings.

Sampling time is optimized to reduce unnecessary measurements and save energy. For large areas, lamps can operate in "islands," with a master lamp controlling local slaves.

4 Results

Simulation results show that the fuzzy controller successfully adapts lamp current according to environmental conditions. Table 1 summarizes approximate energy savings observed during the simulations.

Table 1: Energy savings under different scenarios

Scenario	Energy saving (%)
Clear summer night	25
Thunderstorm	17–20
Foggy night	0

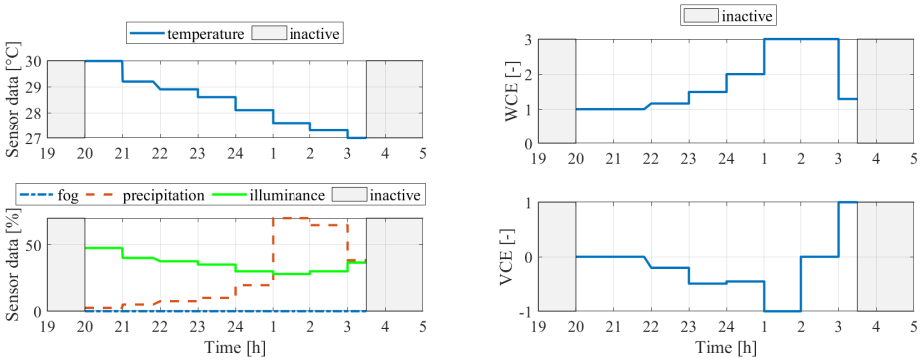
The highest energy savings occur during clear weather conditions when the system reduces lamp current while maintaining sufficient illumination. In contrast, during adverse weather conditions such as fog, higher current levels are required to preserve acceptable visibility.

Figure 2 illustrates the controller behavior during the summer night with a brief period of thunderstorm. During the rainfall period the visibility conditions deteriorate and the fuzzy controller increases the LED forward current in order to compensate for the reduced illumination. After the thunderstorm subsides the controller gradually decreases the current level, returning to a more energy-efficient operating point. This adaptive behaviour demonstrates the capability of the proposed fuzzy system to react to dynamically changing weather conditions.

Overall, the simulation results demonstrate that the fuzzy rule-based controller can achieve energy savings of approximately 20–25% under normal operating conditions. The system remains robust even when sensor values vary or environmental conditions change dynamically.

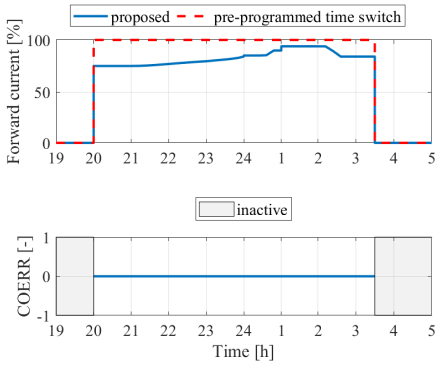
5 Conclusion

A fuzzy rule-based smart street lighting system was proposed and evaluated. The system adapts LED lamp operation to current weather conditions, achieving substantial energy savings while maintaining adequate visibility. Future work may include integration of motion and traffic detection, as well as inclusion of color temperature control for improved visual comfort.



(a) Weather parameters.

(b) WCE and VCE outputs.



(c) Current controller outputs.

Figure 2: Simulation results for the summer night with a brief thunderstorm showing the variation of sensor signals and the resulting adaptive adjustment of LED forward current [5]

Acknowledgements

Project no. 145934 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the K_24 funding scheme. The authors are also grateful for the support of the GINOP_PLUSZ-2.1.1-21-2022-00120 project "Development of fast prototype preparing method to support the design of intelligent LED lighting systems related to the circular economy."

References

[1] Dizon E., Pranggono B. Smart streetlights in smart city: a case study of Sheffield. *J Ambient Intell Hum Comput*, 13(4):2045–2060, 2021.

[2] Elejoste P., et al. An easy to deploy street light control system based on wireless communication and LED technology. *Sensors*, 13(5):6492–6523, 2013.

[3] Lau S.P., Merrett G.V., Weddell A.S., White N.M. A traffic-aware street lighting scheme for smart cities using autonomous networked sensors. *Comput Electr Eng*, 45:192–207, 2015.

[4] Mohandas P., et al. Artificial neural network based smart and energy efficient street lighting system. *Sustain Cities Soc*, 48:101499, 2019.

[5] Brand Á., Magyar A., Hangos K. M. Fuzzy rule-based smart street lighting. *LEUKOS: The Journal of the Illuminating Engineering Society*, 2025.

Prediction of 90-day mRS values of stroke patients based on textual care documentation

Etele Balogh, András Kicsi

Abstract: Acute stroke has many long-lasting side effects which can limit the patient's motor and cognitive functions. During the after care of such cases, there is a great need for an objective measurement scale, which is equipped to reliably track the patient's condition. The Modified Rankin Scale (**mRS**) is a widely used method to represent the patient's condition after the initial stroke case. In this paper we will focus on the information content of the provided Hungarian textual nursing documentation, and will try to answer, based on our observations, if we can possibly make a reliable solution to predict the individual patients' 90 to 110 day **mRS** scores, based on the available, anonymized, textual, medical reports.

Keywords: stroke, mRS, text-based, prediction, Hungarian documentation

1 Introduction

The **mRS** value is on a 7 point scale, which aims to describe the patient's physical state, the level descriptions are detailed in Figure 1. The baseline **mRS** score can be highly subjective and a lengthy process to determine. To address these issues researchers developed multiple different methods, such as the modified Rankin Scale Structured Interview (**mRS-SI**) [1] which has cut the grading time and using this method, the consensus among raters increased. Later, Bruno A. et al. [2] developed the simplified Modified Rankin Scale questionnaire (**smRSq**). This method was able to produce nearly the same reliability metrics meanwhile shortening the interview's timespan and providing easily understandable questions with only yes or no answers. For the structure of the **smRSq**, see Figure 2. In this paper, we propose an artificial intelligence based method which is based on the **smRSq**, and explore the feasibility of it.

2 Background

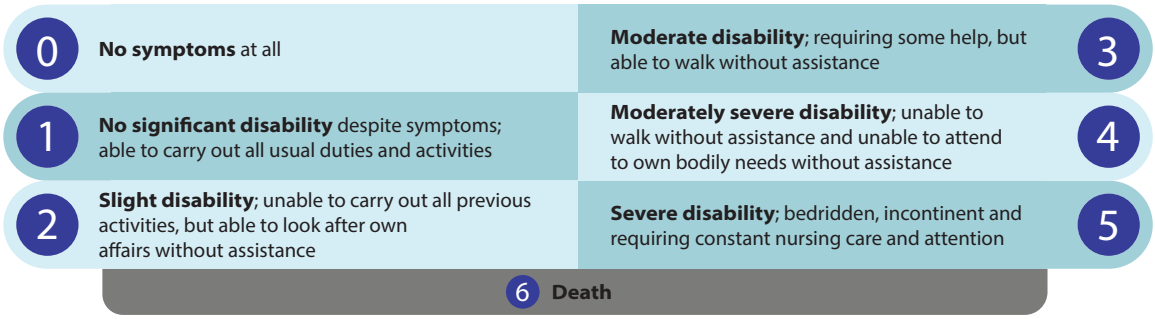


Figure 1: The **mRS** scale with basic level descriptions [3].

Earlier machine learning solutions used simple text representation, researchers usually used unstructured or partly structured textual data, such as different therapy documentation, discharge papers etc. The research efforts that have achieved good results [4, 5] usually use datasets with thousands of records. Sung et al. [6] achieved approximately *0.79* accuracy on the 3-month prediction of the patient's **mRS** score. On a relaxed version of the problem (*good* - **mRS** score of *0-2* - and *bad* - **mRS** score of *3-6* - outcomes), Sung et al. [5] achieved *0.77-0.81* AUC based on raw textual records, while Fernandes et al. [4] were able to achieve *0.59* F1-score

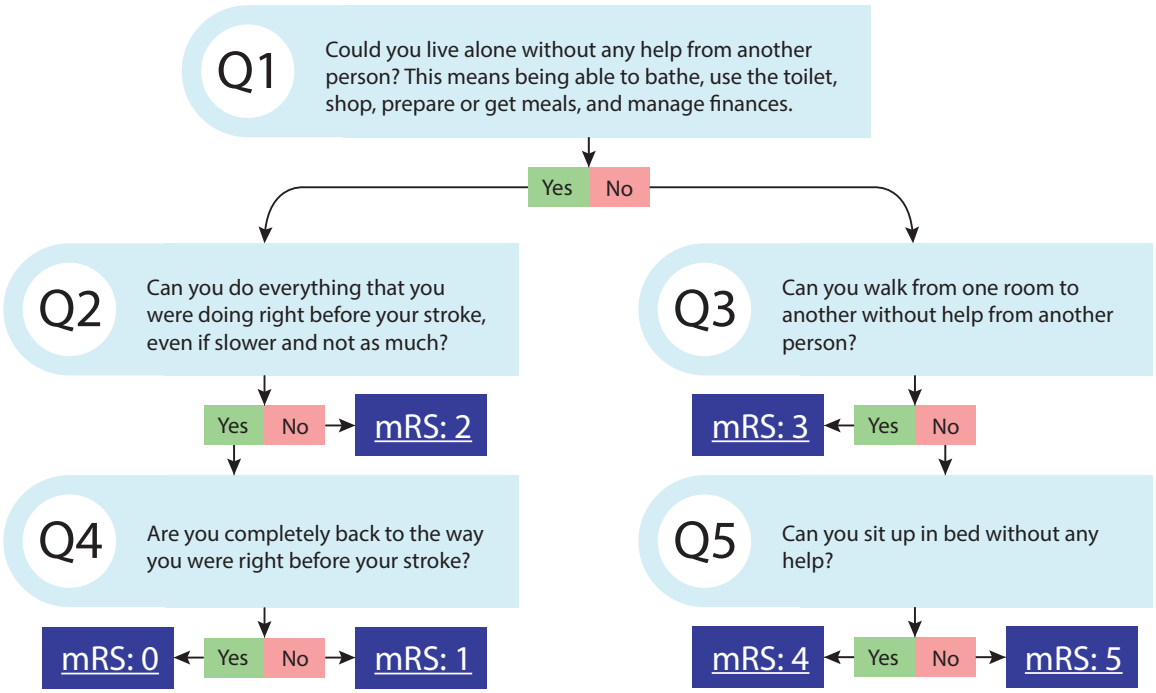


Figure 2: The simplified **mRS** questionnaire algorithm [2].

on the *7*-label **mRS** prediction problem. To our knowledge, AI-based mRS scoring has not been tackled in Hungarian before.

Based on both consultations with medical professionals and related research, we decided, that the label **6** is less worthwhile to assess automatically. After death, there is often no accessible medical documentation which could be relevant for **mRS** prediction (at least not often in the hands of the professionals who deal with stroke cases, especially since patient death can occur due to a variety of non-stroke-related reasons in the 3 month time), thus we amended this level from our current experiments. With the help of the aforementioned medical professionals, we determined the relaxed problem of the 90 day prediction of 6-label classification (from *0* to *5* **mRS** score) which categorizes patients' 90-day **mRS** score into "*good*" (**mRS** score from *0* to *2*) and "*bad*" (**mRS** score from *3* to *5*) outcomes, similarly to [5], reducing the search space to a binary prediction problem.

3 Dataset

For our experiments, a dataset was provided by the *University of Pécs* which was composed of real patients' Hungarian textual documentation. The dataset contained **130** patient's stroke related textual documentation, totalling to **6697** textual records, separated into "*visits*". These records were mixed, but the vast majority of them unstructured text with some of the patients having structured records too.

We annotated *37* different patients **50** unique randomly selected visit documentations from the dataset with the help of a linguist annotator. When the annotator wasn't able to give a sure answer for a question, based on the documentation then she could rely on the answers given to previous question in determining it. This annotation used the **smRSq** algorithm questions and it contained the answer (based on the available documentation) for every question based on only the text, if it was discernable. We note that the annotator expressed the hardness of the task because the documentation simply did not provide exact answers in most cases. The 50 records had the accumulated token count of **69513**.

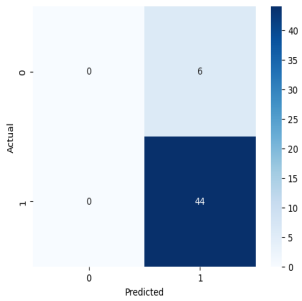
4 Experiments

We provide a proposal and initial, very small-scale results of a method based on the **smRSq** algorithm. Our method uses Large Language Models (referred to as **LLM**) to answer the given questions based on the provided documentation. Our chosen base model was the PULI LLUMIX [7] which was (at the time of the experiments) state-of-the-art Hungarian **LLM**.

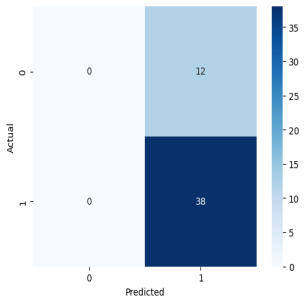
We fine-tuned 5 different models using 10-fold cross-validation, each one of them concentrated on one question, which was extracted from the **smRSq**'s description (Figure 3). After that, one patient's records were evaluated by all small binary classificators models and the resulting five-dimensional binary vector was evaluated using the boolean expression constructed from the **smRSq**'s algorithm decision tree. Table 1 contains the *minimun*, *maximum* and *mean* accuracy scores of the question answering models and the **smRSq** algorithm's accuracy was **0.7400**. While the accuracy is not low, it is visible that the model most often strayed to always giving the same answer. This behaviour could be explained by data noise, the real information content, and small amount of the training data. [1]

model	**min**	**max**	**mean**
Q1	*0.6462*	*0.7182*	*0.6844*
Q2	*0.6470*	*0.7168*	*0.6845*
Q3	*0.6459*	*0.7039*	*0.6792*
Q4	*0.6383*	*0.7147*	*0.6750*
Q5	*0.6553*	*0.7078*	*0.6735*

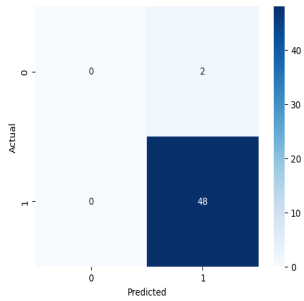
Table 1: Model accuracies



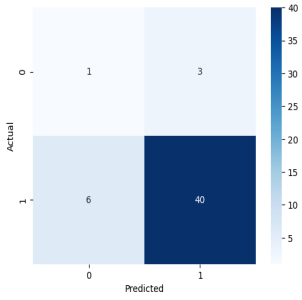
(a) Q1 confusion matrix



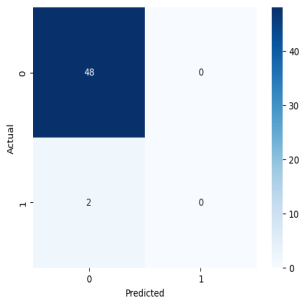
(b) Q2 confusion matrix



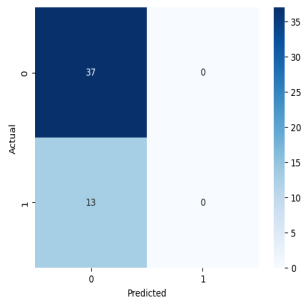
(c) Q3 confusion matrix



(d) Q4 confusion matrix



(e) Q5 confusion matrix



(f) Good/bad confusion matrix

Figure 3: The **smRSq** algorithm confusion matrices of our experiments

[1]The questions (in Hungarian) and the prompt used during the training are available in the online appendix, found at: `https://github.com/Yndiliadrin/CsCs26-mRS-prediction`

5 Conclusion

Our paper presented a question answering approach to automatic **mRS** determination problem. While our early proof of concept experiments show a distinct lean towards a single answer, with sufficient quantity and quality of annotated data, the method could become a well-explainable deep learning based use of **smRSq**. While it is a hard task even for human annotators to determine **mRS** values from documentation only, **smRSq** is used widely in clinical practice. Automatic determination of **mRS** scores can later be used training AI-based solutions in assessments of the results of various medical treatments or the need for them in individual cases.

Acknowledgements

We extend our gratitude to the colleagues who provided data and valuable expertise to the experiments from the University of Pécs and the National Laboratory of Translational Neuroscience.

Disclosures

Generative Artificial Intelligence was used to spell check the current paper for linguistic errors.

References

[1] Wilson, J. T. L., Hareendran, A., Grant, M., Baird, T., Schulz, U. G. R., Muir, K. W., & Bone, I. (2002). *Improving the assessment of outcomes in stroke: use of a structured interview to assign grades on the modified Rankin Scale.* Stroke, 33(9), 2243-2246.

[2] Bruno, A., Shah, N., Lin, C., Close, B., Hess, D. C., Davis, K., Baute, V., Switzer, J. A., Waller, J. L., & Nichols, F. T. (2010). *Improving modified rankin scale assessment with a simplified questionnaire.* Stroke, 41(5), 1048-1050.

[3] Nobels-Janssen, E., Postma, E. N., Abma, I. L., van Dijk, J. M. C., Haeren, R., Schenck, H., Moojen, W. A., den Hertog, M. H., Nanda, D., Potgieser, A. R. E., Coert, B. A., Verhagen, W. I. M., Bartels, R. H. M. A., van der Wees, P. J., Verbaan, D., & Boogaarts, H. D. (2022). *Inter-method reliability of the modified Rankin Scale in patients with subarachnoid hemorrhage*. Journal of Neurology, 269(5), 2734-2742.

[4] Fernandes, M. B., Valizadeh, N., Alabsi, H. S., Quadri, S. A., Tesh, R. A., Bucklin, A. A., Sun, H., Jain, A., Brenner, L. N., Ye, E., Ge, W., Collens, S. I., Lin, S., Das, S., Robbins, G. K., Zafar, S. F., Mukerji, S. S., & Brandon Westover, M. (2023). *Classification of neurologic outcomes from medical notes using natural language processing*. Expert Systems with Applications, 214.

[5] Sung, S. F., Hsieh, C. Y., & Hu, Y. H. (2022). *Early Prediction of Functional Outcomes After Acute Ischemic Stroke Using Unstructured Clinical Text: Retrospective Cohort Study*. JMIR Medical Informatics, 10(2)

[6] Sung, S. F., Chen, C. H., Pan, R. C., Hu, Y. H., & Jeng, J. S. (2021). *Natural language processing enhances prediction of functional outcome after acute ischemic stroke.* Journal of the American Heart Association, 10(24), e023486.

[7] Győző, Y. Z., Réka, D., Gergő, F., Péter, H., Enikő, H., Mariann, L., Noémi, L.-N., Gábor, M., Bence, S., Kristóf, V., Tamás, V., & Tamás, V. (2025). *PULI LlumiX modell: egy folytatólagosan előtanított nagy nyelvi modell.*

Implementation and computational verification of digital signal processing pipelines for temporal light artifact measurements

Róbert Nagy

Abstract: The widespread adoption of LED lighting and high-frequency pulse-width modulation (PWM) dimming has introduced significant challenges in maintaining light quality, specifically regarding temporal light artifacts (TLAs) such as flicker and stroboscopic effects. Quantifying these effects requires complex digital signal processing (DSP) models that simulate the human bio-perceptual response. This paper presents the implementation and algorithmic verification of an open Python pipeline that calculates the short-term flicker indicator (P_{st}^{LM}) and the stroboscopic effect visibility measure (SVM) in compliance with IEC TR 61547-1[1] and IEC TR 63158[2]. To validate the software implementation, a high-precision reference radiator system was built and utilized to generate a standardized test suite of 26 waveforms with known analytical values. Furthermore, this study evaluates the impact of hardware-level constraints on measurement stability. Experimental data demonstrates that while the algorithmic implementation is robust, the precision of TLA metrics is highly sensitive to hardware-induced signal distortion and quantization errors. Specifically, strict thresholds are defined for analog-to-digital converter (ADC) resolution and sampling rates, providing a verified benchmark for developing accurate, software-defined measurement instrumentation.

Keywords: DSP, algorithmic verification, TLA, data acquisition, human-computer interaction.

1 Introduction

The rapid global transition from incandescent and fluorescent lighting to Light Emitting Diode (LED) technology represents a fundamental shift in how artificial light is generated and controlled. Unlike legacy sources, LEDs are capable of near-instantaneous response to changes in driving current. To achieve efficient dimming, manufacturers often employ Pulse Width Modulation (PWM) or high-frequency switching drivers. While energy-efficient, these digital control methods frequently introduce unintended high-frequency modulations in light output, leading to temporal light artifacts (TLAs). The requirement for flicker-free operation is particularly critical in the domain of intelligent street lighting and transportation infrastructure. With the advent of autonomous driving and camera-monitor systems (CMS), the mitigation of TLAs is no longer solely a human-centric issue, but also a critical factor for machine vision reliability. Beyond the stroboscopic effect, which affects the perceived motion of objects, modulated LED lighting can trigger the Phantom Array Effect (PAE)[3]. This visual artifact occurs during rapid eye movements (saccades), causing stationary or moving light sources to appear as a series of separate "ghost" images or dashed trails. In nighttime driving scenarios, PAE from street luminaires or vehicle taillights creates significant visual clutter, leading to spatial disorientation, discomfort, and a documented increase in visual response times.

2 Algorithmic background and implementation

TLAs are categorized primarily into flicker (static observer, static light) and the stroboscopic effect (moving observer or object)[4]. Beyond affecting perceived light quality, these artifacts have documented physiological impacts, including increased mental fatigue, migraines, and, in extreme cases, the triggering of photosensitive epileptic seizures. Consequently, the ability to accurately quantify these signals is a critical requirement for human-centric lighting environments. Historically, the lighting industry relied on simple time-domain metrics such as

Modulation Depth (MD) and Flicker Index (FI). However, these metrics are inadequate for modern frequency-modulated lighting because they fail to account for the frequency-dependent sensitivity of the human visual system. To address this, international standards bodies introduced complex bio-perceptual models. The P_{st}^{LM} (short-term flicker indicator) algorithm uses a multi-stage filter chain to simulate the cornea-retina response and a statistical classifier to model brain-level perception. The SVM (stroboscopic effect visibility measure) utilizes frequency-domain analysis and Minkowski summation to predict the visibility of stroboscopic effects in the 80 Hz to 2000 Hz range. The high-frequency nature of PWM signals necessitates high-speed data acquisition (DAQ) and high-resolution analog-to-digital converters (ADCs) to avoid aliasing and quantization errors. Because researchers are often forced to rely on "black-box" commercial hardware, this paper develops a transparent Python-based framework for TLA signal processing, benchmarking it against hardware constraints to ensure reproducibility.

3 System architecture and methodology

The hardware architecture was designed to meet the strict sampling and resolution requirements of the IEC standards (Figure 1). The signal chain consists of a $V(\lambda)$-corrected photodetector converting illuminance into current. For the analog processing layer, a Keithley 6485 picoammeter was utilized as a transimpedance amplifier (TIA) with an integrated low-pass filter (approx. 1400 Hz cut-off) to prevent aliasing. Data capture was performed using a Tektronix MSO 5 oscilloscope (12-bit ADC) and a Keithley DMM7510 high-precision digital multimeter (18-bit ADC), with raw time-series data exported for offline processing. To verify the algorithmic implementation, a hardware-in-the-loop "Reference Radiator" was constructed.

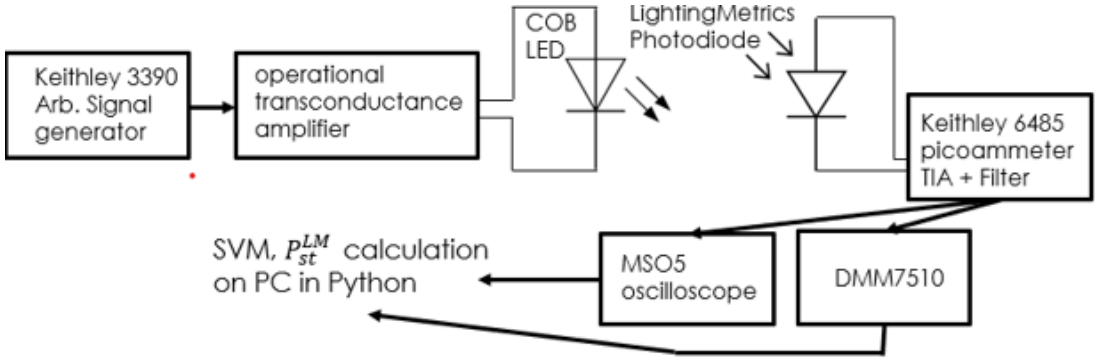


Figure 1: Measurement setup.

A Keithley 3390 arbitrary signal generator was used to drive a Philips COB LED via an operational transconductance amplifier (OTA), generating mathematically defined signals where the TLA metrics could be calculated analytically. The computational framework was developed as a modular Python library utilizing the SciPy and NumPy stacks. By utilizing an open, software-defined architecture, this pipeline can be seamlessly integrated into automated testing environments or adapted for real-time edge computing. To accommodate heterogeneous hardware environments, a dedicated parser module abstracts hardware-specific file formats into standardized arrays. The P_{st}^{LM} calculation follows a digital filter chain, mapping the standard's s-domain transfer function into z-domain infinite impulse response (IIR) filter coefficients via bilinear transformation. This signal chain sequentially applies: AC-coupling and normalization, bio-perceptual filtering (Butterworth filters modeling cornea-retina response), non-linear demodulation (squaring the signal and applying a 300 ms low-pass filter), and statistical post-processing (binning the time-series into 10,000 classes to derive the final value via a root-sum-square of specific percentiles). The Stroboscopic Visibility Measure (SVM) implementation utilizes a frequency-domain approach. To optimize computational efficiency, the module

employs a next-power-of-2 padding strategy for the Fast Fourier Transform (FFT). A Hanning window is applied to the time-domain signal to reduce spectral leakage, with a compensation factor applied to maintain amplitude accuracy. The Fourier coefficients are scaled by a sensitivity curve (`getSensitivityStrobo`) modeled from a custom exponential-logistic function. The implementation identifies spectral peaks and performs a Minkowski summation with a constant of n=3.7, as defined by the standard equation:

$$SVM = \left(\sum_i \left(\frac{C_i}{T_i} \right)^n \right)^{\frac{1}{n}}$$

where C_i is the relative amplitude of the i-th Fourier component, and T_i is the visibility threshold for the stroboscopic effect at the corresponding frequency. This non-linear aggregation effectively models the human perception of multiple simultaneous modulation frequencies. To ensure the software's reliability, a synthetic signal generator (`CustomSignalGenerator.py`) was developed. This allows the system to generate waveforms (Sine, Square, and PWM) with mathematically defined TLA values. These digital signals are used to verify the software's internal logic before applying it to physical sensor data, ensuring that any observed errors in the physical setup are due to hardware noise or quantization rather than algorithmic flaws.

4 Experimental results and discussion

The experimental phase focused on two distinct goals: validating the Python algorithm's computational accuracy against industry-standard benchmarks and evaluating the physical performance limits of different data acquisition (DAQ) hardware configurations. To validate the algorithmic logic, the Python implementation was tested against 26 standardized reference waveforms provided by Signify. To ensure high fidelity during this digital verification, ideal P_{st}^{LM} waveforms were synthesized at 5 kS/s spanning 180 seconds, while ideal SVM waveforms were generated at 1 MS/s for 1 second. The Python software achieved excellent accuracy, with the vast majority of P_{st}^{LM} and SVM calculations showing a deviation of less than 5 % compared to the MATLAB reference values. Larger relative errors (e.g., 20–33 %) were strictly isolated to waveforms where the absolute metric values were exceptionally low (e.g., SVM $<$0.05), meaning the variations were mathematically magnified but practically negligible as they fell well below the threshold of human perception and standardized noise limits. A comparative analysis of physical measuring instruments revealed that ADC resolution and sampling frequency are the primary bottlenecks for TLA measurement stability. The standardized noise floor limits are defined as $P_{st}^{LM} < 0.1$ and SVM < 0.05. While the P_{st}^{LM} metric is computationally stable at sampling rates as low as 5 kS/s, the SVM calculation requires significantly higher rates to accurately capture high-frequency PWM components without aliasing. Several hardware setups were evaluated, including a Tektronix MSO 5 oscilloscope (12-bit ADC, oversampled to 16-bit at 1 MS/s) and a Keithley DMM7510 digital multimeter (18-bit ADC). Testing revealed that the Keithley 6485 picoammeter used as a transimpedance amplifier in the signal chain was the weakest link; it picked up excessive low-frequency noise that skewed P_{st}^{LM} baseline readings and introduced severe signal distortion at frequencies above 400 Hz, which corrupted the SVM frequency-domain analysis. The most accurate configuration bypassed the active amplifier entirely, combining the photodetector directly with the 18-bit DMM7510 using a passive 500 Ω shunt resistor. This setup yielded a near-zero noise floor and reduced the overall measurement error to < 1.5 %. Finally, the self-built DAQ system was benchmarked against two commercially available portable flickermeters in a complex real-world scenario. The test subject was a 12-channel HILED luminaire operating at an 1133 Hz PWM frequency. Because the light output consisted of highly complex, overlapping harmonic waveforms, traditional time-domain metrics like Modulation Depth (MD) failed entirely to characterize the visual impact.

More critically, the commercial flickermeters failed to process the high-frequency channel overlap, registering massive false-positive P_{st}^{LM} values of 2.118 and 0.783. Compared to the true baseline value of 0.025 captured by the verified setup, these commercial instruments produced catastrophic measurement errors of 8372 % and 3032 %, respectively. Such extreme discrepancies in "black-box" devices typically stem from inadequate hardware anti-aliasing filters or internal signal clipping, which misinterpret high-frequency phase shifts as low-frequency flicker. In contrast, the high-resolution DMM7510 and Python DSP pipeline maintained complete stability, proving that transparent, high-bit-depth, and high-sample-rate instrumentation is necessary for evaluating modern software-controlled lighting environments.

5 Conclusion and future work

This paper presented a comprehensive framework for the digital measurement of temporal light artifacts (TLAs). By treating flicker measurement as a high-fidelity signal processing challenge, critical hardware requirements for modern lighting instrumentation were established. The developed Python library accurately reproduces the results of industry-standard MATLAB models, providing an extensible tool for the research community. Regarding ADC precision, it is concluded that a minimum resolution of 11 bits is required for stable P_{st}^{LM} calculations, while 12-bit or higher is necessary for SVM to maintain a noise floor below 0.05. Furthermore, the SVM requires sampling rates of ideally 20 kS/s or higher to accurately capture high-order harmonics in PWM-driven LEDs. High-precision instruments outperform traditional "black-box" flickermeters by avoiding the high-frequency phase distortion observed in complex luminaire setups. Beyond algorithmic verification, the synthetic generation of these light waveforms serves as a foundation for High-Fidelity Virtual Driving Simulators. To safely evaluate the impact of flickering illumination on human performance, these signals can be computationally modeled within a simulation engine rather than relying on the physical refresh rate of a display. This allows researchers to simulate 'worst-case' lighting scenarios and evaluate their impact on human subjects and machine vision algorithms within a risk-free environment. Future work will explore the integration of these validated TLA models into real-time simulation engines (e.g., Unreal Engine or Unity). Modeling the interaction between high-frequency luminaire flicker and digital camera shutter speeds (the 'rolling shutter' effect) enables the development of more robust computer vision systems immune to artifacts found in LED street lighting. As lighting becomes increasingly integrated into digital infrastructure, standardizing TLA measurement practices through transparent, open-source computational tools will accelerate the development of healthier, safer lighting environments across all application domains.

References

[1] International Electrotechnical Commission, *IEC TR 61547-1: Equipment for general lighting purposes - EMC immunity requirements - Part 1: An objective voltage fluctuation immunity test method*, IEC, 2020.

[2] International Electrotechnical Commission, *IEC TR 63158: Equipment for general lighting purposes – Objective test method for stroboscopic effects of lighting equipment*, IEC, 2018.

[3] Roberts, J. E., & Wilkins, A. J. (2013). "Artifacts and the phantom array effect." *Lighting Research & Technology* 45(1), 87-100.

[4] CIE TN 006:2016. "Visual Aspects of Time-Modulated Lighting Systems – Definitions and Models.", CIE, 2016.

Optimal Object Arrangements for Star Graph Comparison Structures in the Bradley–Terry model

László Gyarmati

Abstract: Paired comparison models are sophisticated tools for information retrieval when specifying a scale value is very uncertain. Previous studies have investigated optimal comparison structures under a fixed number of pairwise comparisons, if the number of compared paired is fixed. Now, for each optimal structure, we investigate the optimal arrangement of the objects within the structure itself—that is, how to place the individual objects within a given structure when we have a prior assumption about their ranking. In our work, we present, among the previously identified optimal comparison structures, – in the stochastic Bradley-Terry model – which object arrangement is optimal in the case of star graphs, that is, which arrangement allows us to extract the maximum possible amount of information.

Keywords: paired comparison; Bradley-Terry model; information retrieval; graph of comparison

1 Introduction

Efficient information extraction is becoming increasingly crucial in today's world. We are applying more and more AI models in these areas with increasing success. AI models perform excellently when processing large amounts of data, but not when only a small amount of data is available; therefore, other mathematical models are needed for information extraction in low-data situations. Paired comparison models are a possible tool for applications in such areas. Using these models data are collected by comparing the objects to evaluate in pairs. Usually data are more reliable if the objects are compared to each other, than if they are characterized by a scale value. The most commonly applied pairwise comparison model is the pairwise comparison matrices (PCM)-based models, including Analytic Hierarchy Process [1]. Another major family of pairwise comparison models is the family of Thurstone-motivated models, which has a probabilistic background. A commonly used variant is the two-option Bradley–Terry model.

Paired comparison models perform particularly well in information extraction when only a small amount of data is available [2]. Previous studies have investigated optimal comparison structures for information extraction in pairwise comparisons under different models, assuming a fixed number of comparisons [3, 4, 5], supposing the number of compared paired was fixed. It has been shown that, by carefully selecting the pairs to be compared, one can obtain estimates closer to the true strengths of the evaluated quantities with fewer pairs than with a poorly chosen but larger number of compared pairs. Moreover, the optimal comparison structures are the same for both PCM-based and Thurstone-motivated models. If the number of objects is n, and the number of compared pairs equals $n-1$, the star graph turns out to be the best structure in the case of $n = 4, 5, 6$. Star graphs are structures in which all other objects are compared to a single central object, with no comparisons among the remaining objects. In this work, we aim to continue this line of research. Specifically, in the case of a star graph structure, we examine which arrangement of the objects allows us to extract the maximum amount of information, assuming we have prior knowledge about the strengths or rankings of the objects. Which object should be in the center of the star? In this work, we investigate this problem with the 2-option Bradley–Terry model via computer simulations.
The structure of this short paper is as follows: after this introduction, Section 2 presents the 2-option Bradley–Terry model, used in our study. In Section 3.1 we introduce the simulation applied in our research, and in Section 3.2 we present the results obtained. In Section 4, we summarize the results and outline possible directions for future research.

2 The investigated model

Paired comparison models with stochastic background assume that the performance of an object during the actual comparison is a random value. Thurstone published his idea in 1927 [6] assuming that the latent random variables are Gauss distributed. The expectation of these random variables are called the strengths which have to be estimated. Two-option models allow "better" and "worse" opinions in decisions. When the decision occurs, the observer decides according to whether the difference of the latent random variables corresponding to the compared objects is greater than, or less than zero. In the case of Bradley-Terry model the distribution of the differences is the logistic distribution [7]. Denoting the latent random variables by ξ_i and their expected value by m_i, $i = 1, \cdots, n$, the following equality holds:

$$\xi_i - \xi_j = m_i - m_j + \eta_{i,j}, i = 1, ..., n, j = 1, ..., n, i \neq j, \tag{1}$$

and $\eta_{i,j}$ are supposed to be independent identically distributed random variables.
Let the matrix of input data be denoted by A, and let the element $A_{i,j,1}$ represent the number of times when decision i is "worse" than j was made comparing objects i and j. Similarly, $A_{i,j,2}$ denotes how many times the "better" decision was made from the perspective of object i when object i was compared with object j. Of course, $A_{i,j,1} = A_{j,i,2}$ for every (i, j) $i \neq j$ pair and $A_{i,i,k} = 0$, $i = 1, \cdots, n$, $k = 1, 2$. Given the distribution of $\eta_{i,j}$, the probabilities of the events $\xi_i - \xi_j < 0$ and $\xi_i - \xi_j \geq 0$ can be computed. Let us denote them by $p_{i,j,k}$, $i, j = 1, \cdots, n$, $k = 1, 2$. The probability of the data matrix A in the function of the expectations, the likelihood function, is

$$L(A \mid m_1, m_2, ..., m_n) = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} (P(\xi_i - \xi_j < 0))^{A_{i,j,1}} \cdot (P(\xi_i - \xi_j \geq 0))^{A_{i,j,2}}. \tag{2}$$

The maximum likelihood estimation of the strength parameter vector is the n dimensional vector $\underline{\widehat{m}}$ for which (2) takes its maximal value. The maximal value is attained and is unique when certain conditions are satisfied [8]. We note that, instead of (2), its logarithm is usually optimized. Moreover, the argument of the optimum does not change if all elements of A are multiplied by the same positive constant. This remark will be used in the simulations.

3 The method of investigations: computer simulation

3.1 Method of simulation

We aim to investigate how to arrange the objects so that the estimated strengths based on pairwise comparisons in the star graph edges are as close as possible to the exact values. We applied computer simulations. The simulations are conducted similarly to those in [3]. The steps of the simulation are as follows:

1. Fix the exact strengths $m_i = (i - 1) \cdot d$, $i = 1, \cdots, n$.

2. Based on these strengths, all $p_{i,j,k}$ $k = 1, 2$ are calculated. These values are set as data, i.e., $A_{i,j,k} = p_{i,j,k}$, $k = 1, 2$. It can be shown that this dataset is consistent, see [3].

3. In [3] it was proved that if the dataset is consistent, both complete and incomplete comparisons would yield the exact strength maximum likelihood estimation, supposing the incomplete dataset can be evaluated. To avoid this, we perturb the dataset $A_{i,j,k}$ to obtain an inconsistent dataset. For each (i, j) pair, $i < j$, we add uniformly distributed random values from the interval $(-l; l)$ to the values $A_{i,j,1}$ and subtract its opposite from $A_{i,j,2}$. Here $0 < l$ is a fixed value called magnitude of the perturbation. The resulting dataset is denoted by $\widetilde{A}$. Now, $\sum_{k=1}^{2} \widetilde{A}_{i,j,k} = 1$. We check $0 < \widetilde{A}_{i,j,k} < 1$, $k = 1, 2$. If this condition is

not met for the (i, j) pair, a new uniformly distribution perturbation value is drawn from $(-l; l)$ until the condition is satisfied.

4. The generated complete and inconsistent dataset is evaluated using maximum likelihood estimation to obtain the estimated strengths $\widehat{\underline{m}}$.

5. Now we compute the estimated parameter values based on the incomplete datasets of each star graph. The estimated parameters is compared to the exact expectations using different metrics. These metrics characterize the effectiveness of information retrieval.

6. Steps 1–4 are repeated for each simulation. The resulting metrics are then averaged across all simulations to obtain the final distances in differences metrics.

There are two main sources of information distortion: perturbed data and incomplete comparisons. We investigated the effects of both sources. We applied 4 metrics in our study: the Euclidean distance between $\underline{m}$ and $\widehat{\underline{m}}$ denoted by EU; the Pearson correlation index between the exact and estimated strengths denoted by PE; as well as the Spearman (ρ) and Kendall (τ) rank correlations between the rankings of exact and estimated strengths. The number of objects were $n = 4, \cdots, 12$. For every magnitude l and for every object number n the same phenomenon was observed.

3.2 Simulation results

In the presented cases, the numbers of object were $n = 11, 12$, the differences in expected values were $m_i - m_{i-1} = d = 0.25$ for $2 \leq i \leq n$, and the perturbation magnitude was $l = 0.1$ and the numbers of edges were $NE = 10$ and 11, respectively. We performed 10^6 simulations. The results we obtained are shown in Tables 1 and 2.

Table 1: Results for star graphs consisting of 11 objects.

index	NE	EU	PE	ρ	τ
1	10	1.622	0.885	0.915	0.795
2	10	1.378	0.913	0.934	0.829
3	10	1.100	0.936	0.948	0.853
4	10	0.947	0.949	0.956	0.868
5	10	0.868	0.956	0.961	0.877
6	10	**0.843**	**0.958**	**0.962**	**0.880**
7	10	0.868	0.956	0.961	0.877
8	10	0.947	0.949	0.956	0.868
9	10	1.101	0.936	0.948	0.853
10	10	1.385	0.913	0.934	0.829
11	10	1.635	0.884	0.914	0.795
12	55	0.286	0.995	0.998	0.992

Table 2: Results for star graphs consisting of 12 objects.

index	NE	EU	PE	ρ	τ
1	11	1.871	0.874	0.909	0.784
2	11	1.646	0.904	0.932	0.821
3	11	1.407	0.927	0.948	0.850
4	11	1.137	0.946	0.957	0.866
5	11	0.995	0.956	0.963	0.878
6	11	**0.933**	**0.961**	**0.965**	**0.884**
7	11	**0.933**	**0.961**	**0.965**	**0.884**
8	11	0.995	0.956	0.963	0.878
9	11	1.137	0.946	0.957	0.868
10	11	1.414	0.927	0.948	0.850
11	11	1.661	0.903	0.932	0.821
12	11	1.893	0.873	0.909	0.784
13	66	0.300	0.996	0.998	0.993

In Tables 1 and 2, the first 11 and 12 rows correspond to consecutive star graphs, where the i-the object is placed at the center. The last row shows the results when the estimated values based on the perturbed data of the complete graph was compared with the original strengths and ranking. We can observe that there is information bias originating from the perturbations (see the last rows) and from incompleteness of the dataset. The best results are highlighted in bold. All four metrics indicate the same optimum. According to all four metrics, the results are best when the elements of medium strength are placed at the center of the star graph. It is clearly

visible that, for all 4 metrics, the improvement exceeds 5% between the best and worst star graph arrangements, and for EU, it even approaches 50%. Consequently, it matters which objects are compared. It is advisable to place a 'medium-strength object' at the center if preliminary information about the objects is available.

4 Conclusions

The simulation results show that, in the case of a small number of compared pairs, additional information can be extracted through an optimal arrangement if prior knowledge about the ranking is available and the center of the star graph is chosen accordingly.
Further investigation of comparison structures from this perspective would be advisable. It would be worthwhile to perform the investigations also for PCM-based models.

Acknowledgement

The research was supported by the University Research Scholarship Programme (EKÖP) of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund. L. Gyarmati thanks the support.

References

[1] Saaty, T. L. (1990). How to make a decision: the analytic hierarchy process. *European Journal of Operational Research*, 48(1), 9–26

[2] Jeon, J. J.; Kim, Y. (2013). Revisiting the Bradley-Terry model and its application to information retrieval. *Journal of the Korean Data and Information Science Society*, 24, 1089–1099.

[3] Gyarmati, L.; Orbán-Mihálykó, É.; Mihálykó, C.; Szádoczki, Z.; & Bozóki, S. (2023). The incomplete analytic hierarchy process and Bradley-Terry model: (In)consistency and information retrieval. *Expert Systems with Applications*, 229(Part B). http://doi.org/10.1016/j.eswa.2023.120522

[4] Tóth-Merényi, A., Mihálykó, C., Orbán-Mihálykó, É., & Gyarmati, L. (2025). Exploring Consistency in the Three-Option Davidson Model: Bridging Pairwise Comparison Matrices and Stochastic Methods. *Mathematics*, 13(9), 1374. http://doi.org/10.3390/math13091374

[5] Szádoczki, Z.; Bozóki, S. 2025. Optimal sequences for pairwise comparisons: the graph of graphs approach. Annals of Operations Research, 353, 1099–1122. https://doi.org/10.1007/s10479-025-06752-z

[6] Thurstone, L. (1927). A law of comparative judgment. *Psychological Review*, 34, 273–286. https://doi.org/10.1037/h0070288.

[7] Bradley, R.A.; Terry, M.E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39, 324–345.

[8] Ford Jr, L. R. (1957). Solution of a ranking problem from binary comparisons. The American Mathematical Monthly, 64(8P2), 28-33.

Design and analysis of incremental clustering algorithms for large dynamic similarity graphs

Dávid Maliga, Levente Buttyán

Abstract: This work focuses on efficiently clustering large-scale, dynamically growing malware similarity graphs that represent malware repositories, with the primary goal of reducing storage requirements for malware samples. As these malware repositories receive new samples every day, their similarity graph is dynamically growing, therefore, requiring a dynamic graph clustering approach. We propose two incremental clustering approaches that process new samples without reclustering the entire graph from scratch every time new samples are added. The proposed methods efficiently support a differential storage scheme, which achieved a nearly 50% reduction in required storage space.

Keywords: incremental clustering, malware similarity, dynamic graphs

1 Introduction

Large malware repositories such as VirusTotal[1], MalwareBazaar[2] and Kaibou Repo[3] are essential resources of the cybersecurity community, providing a comprehensive collection of malware samples for analysis and research purposes. However, efficiently storing the hundreds of millions of samples that they contain presents significant challenges.

To store samples efficiently, one approach is to use differential storage, where only the differences between similar malware samples are stored rather than storing each individual malware sample as a separate file. Since a significant portion of malware is not entirely new but rather modified variants of earlier versions, there is significant overlap between the binary content of malware samples, especially those belonging to the same family. By identifying and grouping similar malware samples together, one can store such groups differentially, reducing storage requirements while still allowing for lossless retrieval of every sample in the repository.

Differential storage methods organize these groups of similar malware samples into a structure known as a delta tree. A delta tree stores the differences between similar samples in a hierarchical structure. The root of the tree is a representative sample, stored as a whole, while the nodes lower down in the tree do not contain the entire sample but only the difference relative to their immediate parent.

One way to find similar malware samples is to use a similarity digest scheme, such as Trendmicro Locality Sensitive Hash (TLSH) [1]. TLSH generates a hash value for a given input such that similar inputs produce similar hash values. TLSH also provides a function to compare similarity hashes and quantifying their difference. One can think of this as quantifying the dissimilarity of inputs, where a smaller TLSH difference means lower dissimilarity, therefore, higher similarity.

By using TLSH, one can efficiently compute the (dis)similarity between large numbers of malware samples, which allows for building a so-called similarity graph. In a similarity graph, each node represents an individual malware sample, and there is an edge between two nodes if their (dis)similarity is above (below) a certain threshold.

Graph clustering algorithms can then be applied to the similarity graph to identify groups of similar malware samples, which can be stored more efficiently using differential storage. However, since malware repositories are dynamic and receive new samples every day, traditional

[1] https://www.virustotal.com
[2] https://bazaar.abuse.ch
[3] https://ukatemi.com/products/kaibou/

static graph clustering algorithms are not efficient. This is because they would require re-clustering the entire graph from scratch every time new samples are added, which is inefficient. This is where incremental clustering can help, allowing for efficiently updating the clusters in the similarity graph as new malware samples are added, without recomputing the entire clustering.

In this work, we propose two incremental clustering algorithms: the Incremental Join Closest Cluster Head (I-JCCH) algorithm and a modified version of I-JCCH with a limited cluster size n (I-JCCH-n). Our algorithms are designed to efficiently update the clusters in the similarity graph as new malware samples are added. We evaluate the performance of these algorithms on a large dataset of malware samples and compare the compression achieved by a differential storage scheme and the time required to retrieve the differentially stored samples.

2 I-JCCH

First, we introduce our I-JCCH clustering algorithm. The main idea behind I-JCCH is to maintain a set of clusters and their cluster heads, where a cluster is a group of malware samples that are stored together differentially and a cluster head is the root of the delta-tree of the cluster. I-JCCH updates the clustering when new samples are presented to it by considering the similarity of the new samples to the actual cluster heads.

An initial set of clusters and cluster heads can be obtained in an early stage, when the number of malware samples and the size of their similarity graph are still manageable, by clustering the similarity graph using a static graph clustering algorithm. For instance, this algorithm may randomly select a node and all its neighbors in the similarity graph to form a cluster, and repeat this step until every sample is assigned to a cluster. The resulting clusters can then be stored differentially, and the roots of the delta-trees of the clusters become the cluster heads.

When a new malware sample N is encountered, I-JCCH identifies the cluster head H that is the most similar to N. If the similarity between H and N is above a certain threshold, then N is added to the cluster of H. Otherwise (i.e., if the new sample is not similar enough to any existing cluster head), N forms a new cluster and becomes its own cluster head. In this way, I-JCCH can efficiently update the clusters as new malware samples are encountered, without having to recompute the entire clustering from scratch.

3 I-JCCH-n

In I-JCCH, a cluster can grow indefinitely as new samples are added, leading to large clusters where adding new samples to the delta-tree and retrieving stored samples can become inefficient. To address this problem, we introduce a variant of I-JCCH that we call I-JCCH-n. The main idea behind I-JCCH-n is to limit the size of every cluster by n.

New samples are typically added in batches. When a batch of new samples is encountered, I-JCCH-n updates the clusters iteratively by processing every new sample in the batch, one after the other. For each new sample N in the batch, I-JCCH-n identifies the closest cluster head, similar to I-JCCH. However, if the cluster of the closest cluster head has already reached its maximum size n, then N cannot be added to that cluster. In this case, I-JCCH-n tries to add N to the cluster of the next closest cluster head. If that fails, then I-JCCH-n tries to add N to the cluster of the next closest cluster head again, and so on. If N cannot be assigned to any existing cluster, it will form a new so-called **virtual cluster** and become a new **virtual cluster head** itself.

Virtual clusters are handled in the same way as actual clusters while processing the batch of new samples. When the entire batch is processed, and every new sample is placed in an actual or a virtual cluster, the actual clusters are updated with their assigned new samples, whereas the virtual clusters are turned into actual clusters by differentially packing them.

This way, I-JCCH-n maintains bounded cluster sizes, thereby improving the efficiency of adding new samples to the delta tree and retrieving stored samples.

4 Results

The dataset used for our measurements was provided by Ukatemi Technologies and comes from the Kaibou Repo. From the dataset, we randomly selected $100,000$ samples, resulting in a subset totaling approximately 116.2 GB in size. This subset was used for all the measurements described in the following.

The proposed methods were evaluated in two scenarios. In the first scenario, we randomly selected $50,000$ samples, applied the static clustering algorithm (described above) to them, and then added the remaining $50,000$ samples incrementally in batches of $1,000$, clustering each batch with the I-JCCH algorithm. The second scenario used the I-JCCH-n algorithm with different cluster size upper-bounds n for our measurements: 25, 50, 100, 200, and 400. To eliminate the random effects of splitting the dataset into two for bootstrapping and incremental processing, as well as the random effects of choosing the batches, we repeated every experiment 10 times and present the results in box-plots. We use two TLSH dissimilarity threshold values for both scenarios: the empirically promising 40 [2] and 86 [1].

First, we examined how efficiently the dataset was compressed. The results are shown in Figure 1. Differentially packing clustered samples reduced the original dataset size by almost 50%. The best compression, 48% was achieved with I-JCCH and a TLSH difference threshold of 86.

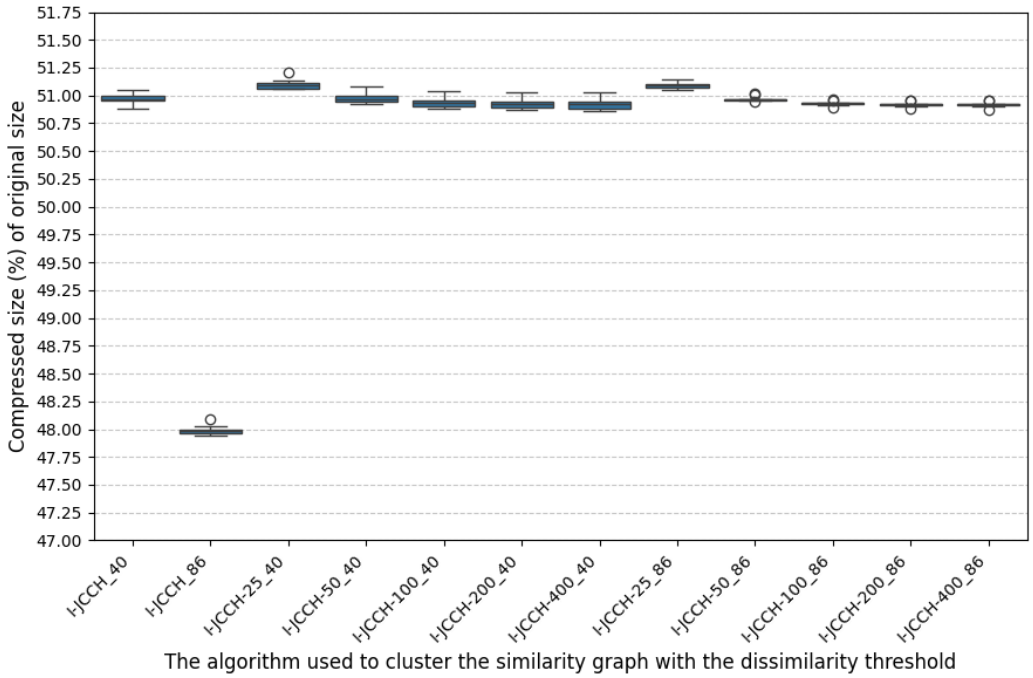


Figure 1: Boxplots of compressed size percentage (relative to original size) for different clustering algorithms and dissimilarity thresholds

Then, we measured the sample retrieval time for each algorithm. For this, we randomly selected $20,000$ samples and measured the time required to retrieve each differentially stored sample. The results are shown in Figure 2. Here, I-JCCH-n achieved the best results, especially with small upper-bounds on the cluster sizes. As it can be seen, the retrieve time of I-JCCH changes significantly when the TLSH difference threshold is changed, while I-JCCH-n remains

stable for every TLSH difference threshold.

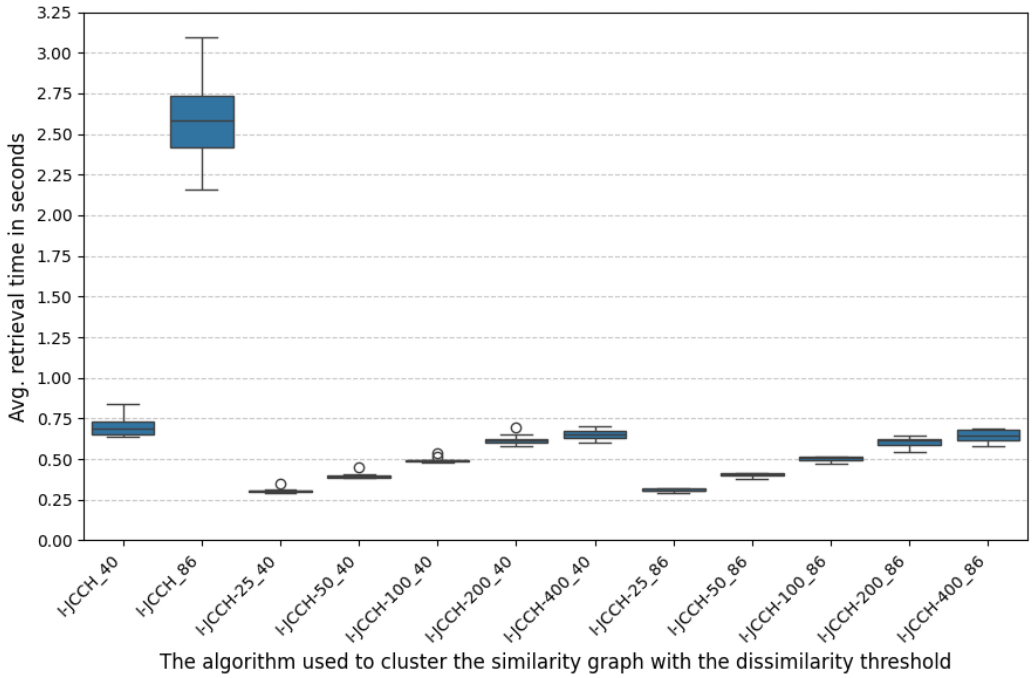


Figure 2: Boxplots of average retrieval time for different clustering algorithms and dissimilarity thresholds

In conclusion, the results show a trade-off between storage efficiency and retrieval time, which is a natural consequence of a higher TLSH dissimilarity threshold. A higher TLSH dissimilarity threshold creates more edges by accepting greater dissimilarities and allowing samples to be clustered with other samples, leading to larger clusters. Larger clusters can improve storage efficiency but also increase retrieval time. At a lower dissimilarity threshold, samples can end up clustered as singletons (clusters containing only one node). In this case, differential storage does not offer any compression benefit, but at the same time, there is no retrieval cost. Selecting the appropriate clustering algorithm and dissimilarity threshold could depend on specific application requirements.

Acknowledgements

The research presented in this paper was carried out in the context of project no. 2023-1.1.1-PIACI_FÓKUSZ-2024-00030 funded by the National Research, Development and Innovation Office of Hungary.

References

[1] J. Oliver, C. Cheng and Y. Chen, "TLSH – A Locality Sensitive Hash," *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, Sydney, NSW, Australia, 2013, pp. 7-13, doi: 10.1109/CTC.2013.9.

[2] L. Buttyán, R. Nagy and D. Papp, "SIMBIoTA++: Improved Similarity-based IoT Malware Detection," *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, Debrecen, Hungary, 2022, pp. 51-56, doi: 10.1109/CITDS54976.2022.9914145.

Technical Challenges in Java Symbolic Execution: Control Flow Graphs and Exception Handling Dynamics

Péter Hajnal, Edit Pengő, István Siket

Abstract: Symbolic execution is a well-established static analysis technique for detecting runtime errors by exploring program execution paths using symbolic inputs rather than concrete values. Despite its theoretical ability to achieve high coverage, its practical applicability is limited by the path explosion problem, where the number of execution paths grows exponentially with program complexity. While existing approaches often focus on constraint solving to mitigate this issue, the accuracy of the underlying Control Flow Graph (CFG) is equally critical, as it determines which execution paths are considered during analysis. This paper presents improvements to the CFG construction in the RTEHunter engine, a Java symbolic executor integrated into the SourceMeter framework. Our approach builds a language-independent CFG from the Abstract Semantic Graph (ASG), leveraging its rich structural and semantic information. We focus on accurately modeling complex Java control flow constructs, including conditional expressions and exception handling mechanisms, which are common sources of imprecision in static analysis. We show that inaccurate handling of exception edges and complex control structures can introduce spurious paths into the CFG, leading to the exploration of infeasible execution branches and an increased number of false positives. To address these issues, we refine the CFG construction to better capture the semantics of Java programs, reducing the number of infeasible paths considered during symbolic execution. To further improve analysis precision, SMT-based path feasibility checking can be applied to eliminate unsatisfiable paths. However, such techniques are only effective when the underlying CFG accurately represents program behavior. These results highlight that precise control flow modeling is a prerequisite for effective symbolic execution in real-world software analysis environments.

Keywords: static analysis, Java, control flow graph, exception handling, path explosion

1 Introduction

Symbolic execution is a powerful static analysis technique for detecting runtime errors by exploring program execution paths using symbolic inputs. However, its practical effectiveness depends not only on constraint solving capabilities, but also on the accuracy of the underlying program representation. In particular, the precision of the Control Flow Graph (CFG) plays a crucial role in determining which execution paths are explored and how accurately program behavior is modeled.

In this paper, we focus on advancing SourceMeter [1], a static analysis framework, by improving the CFG construction used in its Java symbolic execution engine, RTEHunter. SourceMeter supports multiple programming languages and is widely used for detecting code quality issues such as code smells, duplications, and runtime errors. Within this framework, RTEHunter relies on a CFG-based representation of programs to perform symbolic execution, making the correctness and precision of the CFG a central concern. A key aspect of this work is the transformation of the Abstract Semantic Graph (ASG) into a language-independent CFG. The ASG provides a rich, semantically detailed representation of the source code, extending beyond the capabilities of traditional syntax trees. Leveraging this representation enables the construction of a CFG that more accurately reflects program structure and execution semantics.

Despite this, constructing a precise CFG for Java programs remains challenging due to the complexity of language constructs. In particular, conditional expressions, exception handling mechanisms, and implicit initialization patterns introduce non-trivial control flow that is difficult to model accurately. Inaccuracies in representing these constructs may result in the introduction

of infeasible paths, which negatively impact both the performance and the reliability of symbolic execution. To address these challenges, this paper focuses on refining CFG construction in RTEHunter with special attention to complex Java control flow. We improve the handling of conditional expressions to better capture evaluation semantics, and we identify and mitigate issues related to exception handling that frequently introduce spurious paths.

Overall, our work demonstrates that improving CFG construction is essential for enhancing the effectiveness of symbolic execution. By reducing infeasible paths and better aligning the control flow representation with actual program behavior, these improvements contribute to more accurate and scalable static analysis.

2 Advanced CFG Construction for Java Constructs

RTEHunter utilizes the ASG of the program, which is constructed by the analyzer of SourceMeter. The ASG serves as a language-dependent representation of the source code that contains every detail in an internal graph representation [3]. While similar to an Abstract Syntax Tree, it provides additional semantic information essential for deep analysis. From this ASG, RTEHunter builds a language-independent CFG [2].

2.1 Control Flow Architecture and Interprocedural Analysis

In the resulting CFG, each node represents a basic block, which is defined as an abstraction of straight-line program parts guaranteed to be executed sequentially. A jump in the code, such as a conditional statement or a return statement, terminates the current basic block, and the outgoing edges connect to the basic blocks that serve as the targets of that jump. Inside each basic block, the engine maintains the ASG nodes that are visited in a sequential order. Because RTEHunter performs symbolic execution on each method of the program separately, a separate control flow graph is created for all methods. This approach is often a more effective solution for large-scale systems than starting the execution solely from the `main()` method, as it allows for broader code coverage without reaching practical limitations too quickly. Each CFG for a method contains exactly one entry block and exactly one exit block, even if the method contains multiple return statements. The control can only enter and leave the CFG of a method through these two blocks, making them highly useful for handling function calls and returns. During interprocedural analysis, the symbolic engine handles method calls by continuing execution in the called methods and then returning to the original function, mimicking normal execution. Built-in or third-party Java functions are also executed in this manner, though any warnings found within them are filtered from the final output.

2.2 Modeling Complex Java Logic and Initialization Patterns

The complexity of Java's logical expressions requires specific graph structures to maintain accuracy. From a conditional jump, a new basic block starts at each conditional sub-expression. Collector basic blocks are introduced to make handling expressions containing multiple sub-expressions easier. RTEHunter is specifically aware of short-circuit evaluation; for example, if the control reaches a collector block from the true edge of a preceding condition, there is no further branching in that block, and only the true edge is continued. The complexity of Java's logical expressions requires specific graph structures to maintain accuracy. From a conditional jump, a new basic block starts at each conditional sub-expression. Collector basic blocks are introduced to handle expressions containing multiple sub-expressions while preserving their evaluation order. RTEHunter explicitly models short-circuit evaluation; for example, if control reaches a collector block from the true edge of a preceding condition, no further branching is introduced, and only the corresponding edge is continued. This ensures that the CFG faithfully represents

the semantics of compound boolean expressions without introducing spurious control flow paths.

A significant source of false positives arises from issues in the Java analyzer when it fails to correctly interpret certain control-flow constructs, such as exception handling. Improperly resolved exception edges may result in spurious paths in the CFG, causing the symbolic engine to explore infeasible or incorrect branches. Furthermore, initialization patterns in Java can mislead the analysis. Fields initialized outside of a constructor, such as in a separate `init()` method, may appear uninitialized to a symbolic executor if the analyzer is unaware of the design pattern. Similarly, fields injected by a framework, often indicated by annotations such as `@FXML`, will lead to incorrect assumptions and false-positive reports unless the symbolic executor is specifically configured to recognize that these fields are initialized externally.

Originally the CFG for the code snippet above (Figure 1.) was not correctly constructed, as the analyzer failed to resolve the exception edges properly. The `finally` block was not connected when we `return` from the `if` statement seen on Figure 2. A possible solution to this issue is to connect the `finally` block to the `finally` block and the exit block, as shown in Figure 3. However, this introduces spurious paths that are not feasible in actual execution. Our improved CFG, shown in Figure 4, correctly connects the `finally` block to the exit block, while ensuring that the path from the `return` statement. We copied the `finally` block and connected it to the `true` edge of the `if` statement, ensuring that the `finally` block is executed in all cases, while avoiding spurious paths like `code3` seen in Figure 3. This solution seems very simple because the `finally` block is represented by one node only, but usually it is a complex subgraph of the ASG therefore its duplication is a difficult task.

3 Conclusion

In this paper, we improved the CFG generation in the RTEHunter engine by refining the transformation from the Abstract Semantic Graph (ASG) and by addressing key sources of imprecision in Java programs. Our work focused on accurately modeling complex control flow constructs, including conditional expressions, short-circuit evaluation, and exception handling. We showed that improper handling of these constructs can introduce spurious paths into the CFG, leading to incorrect assumptions about program behavior and reducing analysis reliability. By improving the representation of these elements, the constructed CFG more faithfully reflects actual program semantics.

Overall, this work emphasizes that improving CFG construction is essential for achieving accurate and scalable static analysis. Although techniques such as SMT-based path pruning can help reduce infeasible paths, their effectiveness fundamentally depends on the accuracy of the underlying CFG. Without precise control flow modeling, higher-level analysis techniques cannot

```
try {
    code1;
    if (condition) {
        return;
    }
    code2;
} catch (Exception e) {
    // handler block
} finally {
    // finally block
}
code3;
```

Figure 1: Example Java Code

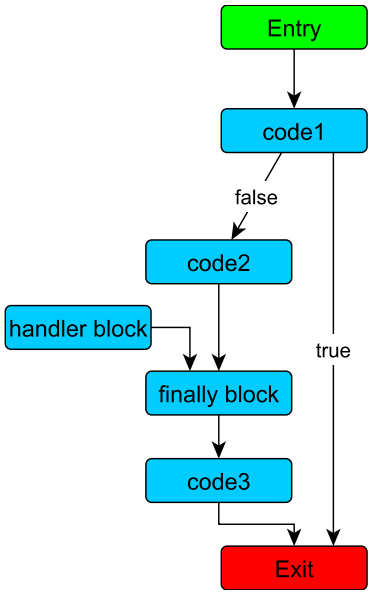


Figure 2: Original CFG

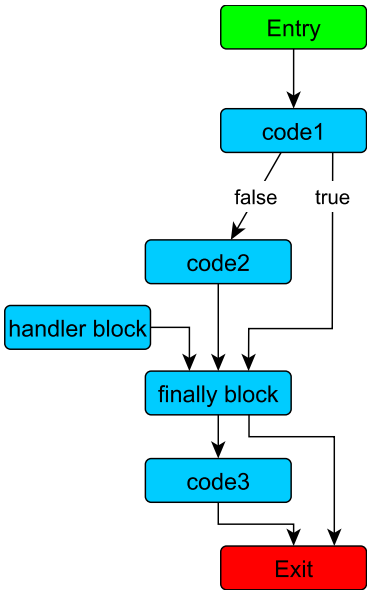


Figure 3: Possible CFG with spurious paths

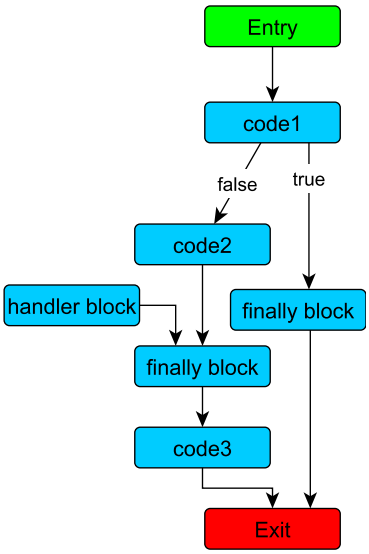


Figure 4: Improved CFG

compensate for structural inaccuracies. Future work may focus on extending these techniques to better support modern Java frameworks and dynamic initialization patterns.

References

[1] Tamás Bakota, Rudolf Ferenc, and Tibor Gyimóthy. Sourcemeter: A source code quality assessment framework. In *27th IEEE International Conference on Software Maintenance (ICSM)*, pages 621–624. IEEE, 2011.

[2] Frances E. Allen. Control flow analysis. In *Proceedings of a Symposium on Compiler Optimization*, pages 1–19. ACM, 1970.

[3] Rudolf Ferenc, Árpád Beszédes, Mikko Tarkiainen, and Tibor Gyimóthy. Columbus - reverse engineering tool and schema for C++. In *Proceedings of the 18th International Conference on Software Maintenance (ICSM)*, pages 172–181. IEEE, 2002.

Reusing Intermediate Model Checking Results With Implicit Abstraction

Dániel Kovács, Milán Mondok

Abstract: Model checking is a formal verification technique that exhaustively explores all possible behaviors of a system to prove correctness or detect errors. Two prominent approaches are Complementary Approximate Reachability (CAR), which simultaneously tightens over- and under-approximations of reachable states, and Counterexample-Guided Abstraction Refinement (CEGAR), which iteratively refines a coarse abstraction until the property can be decided. Implicit abstraction allows wrapping any reachability algorithm in a CEGAR loop, but discards all intermediate results upon each refinement.

In this paper, we propose CARCEGAR, an algorithm that wraps CAR in an implicit CEGAR loop while preserving partial results across refinement iterations. We unify the under-approximation tree of CAR with the abstract reachability graph of CEGAR into a single structure, enabling lazy pruning to retain useful state-space information between iterations. We implemented the algorithm in the open-source Theta model checker and evaluated it on industrial hardware models from the Hardware Model Checking Competition, where it showed promising results.

Keywords: formal verification, model checking, implicit abstraction, CAR, partial results reuse

1 Introduction

Safety-critical computer systems continue to grow in complexity, making exhaustive verification both more critical and more challenging. Model checking addresses this by systematically exploring all reachable states of a system, but the state space grows exponentially with the number of variables, demanding sophisticated symbolic techniques. Among these, Complementary Approximate Reachability (CAR) emerged as a promising yet less extensively studied algorithm that simultaneously tightens over- and under-approximations from two directions, maintaining a *tree of states* from which errors are reachable (the under-approximation) alongside an overapproximation sequence of forward-reachable states.

Counterexample-Guided Abstraction Refinement (CEGAR) offers an orthogonal strategy. Starting from a coarse abstraction, it maintains a tree of discovered abstract states (the *abstract reachability graph*, or ARG) and refines it only when spurious counterexamples are found. The implicit variant of abstraction allows wrapping any reachability algorithm (including CAR) in a CEGAR loop by encoding the abstraction directly into the model. However, this naive combination discards all intermediate results upon each refinement, losing progress.

The core observation behind our work is that CAR's under-approximation tree and CEGAR's abstract reachability graph (ARG) serve analogous roles and can be unified. Moreover, over-approximation frames computed under a coarser abstraction remain sound after refinement, if abstract states were proven unreachable in i steps, no refinement can make them reachable in i steps, since refinement only restricts the transition relation.

We propose CARCEGAR, an algorithm that wraps CAR in an implicit CEGAR loop while preserving partial results. Our contributions are: (1) we unify the under-approximation tree with the ARG, enabling lazy pruning [7] so that only the infeasible suffix of a spurious counterexample is removed upon refinement, (2) we reuse overapproximation frames across iterations, allowing the algorithm to block large abstract state-space regions early, (3) we implement CARCEGAR in the Theta model checker [11] and evaluate it on 317 industrial AIGER benchmarks from the Hardware Model Checking Competition [3], where it solves nearly twice as many instances as naive implicit abstraction CAR.

2 Background

Model checking Model checking [8] is a formal method that systematically explores all possible behaviors of a given input model. Based on the information collected during this exhaustive analysis, it can be determined whether the model is *safe* or *faulty*. In our work, we focus on *reachability* (safety) problems, which investigate whether any of the *error* states are reachable from the *initial* state of the model. As input, we assume a *symbolic transition system* (STS [12]) that consists of 3 SMT [9] formulas, which symbolically encode (1) the initial state of the model, (2) the error states of the model, and (3) the transitions a model can take between different states.

CAR Complementary Approximate Reachability (CAR) [10] is a promising yet less extensively studied (compared to established approaches such as IC3/PDR [4]) hardware model checking algorithm. It maintains both over- and under-approximations of the reachable states: the *over-approximation* bounds the set of states the system could potentially reach from the initial state, while the *under-approximation* captures states from which the error states are definitely reachable. These two approximations are tightened incrementally in a complementary fashion until the algorithm can either confirm that no unsafe state is reachable (proving safety) or construct a concrete trace leading to an unsafe state (proving the existence of a violation).

The CAR algorithm maintains two data structures: an overapproximation sequence $O_0, O_1, \ldots$ of the forward-reachable states and an underapproximation U of the backward-reachable states (i.e., states from which an error state is reachable). Each O_i overapproximates the set of states reachable from the initial state in at most i steps. It is guaranteed to contain every such state, but may also contain spurious ones. The underapproximation U is stored as a tree rooted at the error states, where each node's parent is reachable from it in one step of the model. This means that every state in the tree can reach an error state.

The algorithm iteratively refines these approximations. When it finds a state in O_i, from which a state in U is reachable, then it adds the state to the U tree, and attempts to cut this state out of O_i, by proving that it cannot be reached from the initial state in i steps. If instead, it turns out that this state is reachable from the initial state, then the model is proven unsafe. The model is proven safe when the algorithm finds two consecutive overapproximations that contain exactly the same states ($O_i = O_{i+1}$), and the U states are not reachable from them, establishing a fixed point.

CEGAR Counterexample-Guided Abstraction Refinement (CEGAR) [6, 13] is a widely adopted technique that can formally verify various systems, including both software and hardware. Predicate abstraction only tracks the truth values of logic predicates over the variables instead of concrete variable values. CEGAR starts with a coarse abstraction of the system, which is an overapproximation of the behaviour of the concrete model. It then iteratively refines the abstraction until a sufficient level of precision is achieved.

If the error states are not reachable in the abstract model, then the original system can be considered safe due to the overapproximation, and the algorithm terminates. However, if there is a path between the initial state and the error states in the abstract model, then it must be examined to determine if it is a valid or a spurious counterexample. If the abstract counterexample is feasible, then it means that the model is unsafe and the algorithm terminates. On the other hand, if the path is spurious, then the abstract model needs to be refined. This loop of abstraction and refinement continues until an appropriate precision level is found, where the property can be proven or refuted with certainty.

Reachability-graph-based implementations of CEGAR maintain a so-called *abstract reachability graph* (ARG) [6] to store the abstract states discovered during state space exploration. This is a tree structure, with nodes representing abstract states (each corresponding to $>= 1$ concrete

states) and edges representing transitions between them. The so-called *cover* edges represent that one already expanded abstract state contains all concrete states of another abstract state (one state "covers" the other state), meaning traversal can stop on this branch as all behavior reachable from this state has already been explored. The root of the tree is the initial state of the system, and the tree is expanded in each iteration and pruned (partially or completely) whenever the precision is refined. Lazy pruning [7] (only pruning the infeasible suffix of the spurious counterexample) allows the algorithm to keep partial results between iterations, thus the state space exploration does not have to be restarted from the initial state in each iteration.

3 Our approach: CARCEGAR

Besides the reachability-graph-based approach, the CEGAR loop can also be used *implicitly* [5], which allows wrapping any reachability algorithm that is agnostic to the abstraction (for example IC3 [4] or CAR) in a CEGAR loop. In this approach, the model itself is modified to encode the abstract state space. This is achieved by adding a new Boolean variable for each predicate, which has the same truth value as the predicate. Only these Boolean variables are kept between steps, the exact value of the original variables is "forgotten". However, this approach has a major drawback: the algorithm has to be restarted from scratch every time the model is refined, meaning partial results and progressions are not kept between iterations.

To combine the advantages of the CAR and CEGAR algorithms, we propose the CARCEGAR algorithm. The key idea is simple: wrap the CAR algorithm in an implicit CEGAR loop, but adapt the abstract reachability tree of graph-based CEGAR in order to keep partial results between iterations and speed up the algorithm. Since both algorithms already maintain tree-structured representations of explored states (the underapproximation tree U in CAR and the ARG in CEGAR), unifying them into a single structure allows partial results to be preserved across refinement iterations. This unification also enables the use of established ARG techniques, namely lazy node pruning and cover edges. The algorithm also saves the overapproximations between iterations.

Figure 1 presents an overview of the algorithm. Similarly to CAR, the algorithm maintains an overapproximation sequence. In each iteration (Step 1), it searches the outermost overapproximation for states from which nodes already present in the tree are reachable, and adds these (abstract) states to the tree (Step 2). It then attempts to prove that these states are unreachable from the initial state within a bounded number of model steps, removing them from the overapproximation if successful. If, instead, a complete path from the initial state is found (Step 3), a CEGAR refinement check is triggered: either the counterexample is concretized, proving the model unsafe (Step 6), or it is deemed spurious, in which case the abstraction is refined and the tree is pruned (Steps 4-5). The refined overapproximations and the pruned tree are preserved between iterations, and the next iteration resumes from Step 1. The process continues until either a concrete counterexample is found, or a fixed point is reached, where two consecutive overapproximation frames coincide, establishing that no error state is reachable.

Another major advantage of CARCEGAR over CAR is its ability to eliminate more general sets of abstract states from overapproximations. When the algorithm refines an overapproximation by removing an abstract state, it has proved that this (potentially large) set of concrete states is unreachable from the initial states within a certain number of steps. This unreachability is monotone with respect to refinement: if a state is unreachable in i steps under a coarse abstraction, it remains unreachable in i steps under a finer abstraction. Because CARCEGAR begins with broad abstract states, a single successful cut can rule out a large region of the state space at once. CAR, by contrast, must block states individually, which can be considerably slower.

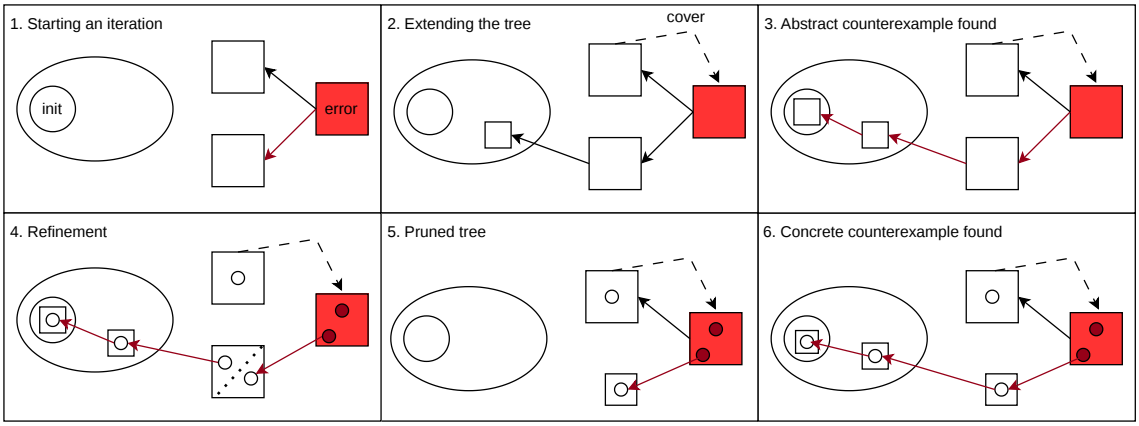


Figure 1: Overview of our approach. The ovals on the left represent the overapproximations O_i. The squares and smaller circles on the right represent the abstract and concrete states, respectively. Step 5 highlights the key insight of our algorithm: lazy pruning allows the reachability tree to be retained between iterations.

4 Evaluation

In order to evaluate our approach, we implemented the CARCEGAR, CAR, and naive implicit abstraction CAR algorithms in the open-source Theta [11] model checker, using identical data structures to ensure a fair comparison. To ensure reliable and reproducible measurements, we used the BenchExec [1] tool, with a time limit of 15 minutes for each run. We evaluated the algorithms on the 317 AIGER [2] models from the bit-level safety category of the 2017–2024 editions of the Hardware Model Checking Competition [3], comprising highly complex, industrial hardware models.

Algorithm	#Solved	#Solved safe	#Solved unsafe	#Unsolved
CAR [10]	32	28	4	285
CAR [10] + implicit abstraction [5]	27	24	3	290
CARCEGAR (our approach)	**62**	**54**	**8**	**255**

Table 1: The number of models solved by each algorithm in the allocated timespan.

The results can be seen in Table 1. The measurements indicate that our CARCEGAR algorithm performed better on this benchmark set than both the CAR and the naive implicit CAR algorithms, solving the largest number of models. This indicates that our approach of retaining the partially pruned reachability tree between iterations can improve the performance compared to the simple implicit approach.

5 Conclusion

Our primary goal during this work was to adapt established techniques from reachability-graph-based CEGAR to the CAR algorithm. The algorithm we presented allows us to wrap the CAR algorithm in an implicit CEGAR loop, but while also retaining partial results between iterations and thus speeding up the algorithm.

Our measurements indicate that the CARCEGAR algorithm performed better on this benchmark set compared to the CAR and implicit CAR algorithms, solving the largest number of models. This suggests that using techniques from CEGAR had a major positive effect on the algorithm's runtime.

Acknowledgments

The computational resources used in this research were provided by the BME VIK CIRCLE cloud infrastructure, which is developed and operated by BME IK and BME IIT, partially funded by faculty resources.

References

[1] D. Beyer, S. Löwe, and P. Wendler. Reliable benchmarking: requirements and solutions. *International Journal on Software Tools for Technology Transfer*, 21(1):1–29, 2019.

[2] A. Biere, K. Heljanko, and S. Wieringa. AIGER 1.9 and beyond. Technical Report, FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University, Linz, Austria, 2011.

[3] A. Biere, N. Froleyks, and M. Preiner. The hardware model checking competition 2024. In *Proceedings of the 24th Conference on Formal Methods in Computer-Aided Design (FMCAD 2024)*, 2024.

[4] A. R. Bradley. Understanding IC3. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 1–14, Springer, 2012.

[5] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. IC3 modulo theories via implicit predicate abstraction. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 46–61, Springer, 2014.

[6] Á. Hajdu and Z. Micskei. Efficient strategies for CEGAR-based model checking. *Journal of Automated Reasoning*, 64(6):1051–1091, 2020.

[7] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2002)*, pages 58–70, 2002.

[8] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem. *Handbook of Model Checking*. Springer Publishing Company, 1st edition, 2018.

[9] D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[10] J. Li, S. Zhu, Y. Zhang, G. Pu, and M. Y. Vardi. Safety model checking with complementary approximations. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 95–100, IEEE, 2017.

[11] T. Tóth, Á. Hajdu, A. Vörös, Z. Micskei, and I. Majzik. Theta: a framework for abstraction refinement-based model checking. In *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design*, pages 176–179, 2017.

[12] Á. Hajdu, T. Tóth, A. Vörös, and I. Majzik. A Configurable CEGAR Framework with Interpolation-based Refinements. In *Formal Techniques for Distributed Objects, Components and Systems*, Lecture Notes in Computer Science, volume 9688, pages 158–174. Springer, 2016.

[13] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, Lecture Notes in Computer Science, pages 154–169. Springer, 2000.

Blockchain-based RAG Architecture for Vulnerability Assessment

Csaba Nagy, Andrea Huszti, Norbert Oláh

Abstract: Nowadays, improving the security of IT systems is a prominent topic. However, vulnerabilities and cyber threats are becoming increasingly advanced. Therefore, the security controls defined by risk management processes must be equally scalable and automated. This paper presents a novel security architecture integrating automated vulnerability scanners (OWASP ZAP, Nessus, SQLMap) with an LLM-driven AI module. AI automates findings summarization, CVSS risk scoring, STRIDE threat analysis, and generating actionable remediation suggestions. To mitigate hallucinations the system employs a multi-model collaboration strategy (GPT-4 and Gemini 2.5 Pro) paired with a Human-in-the-Loop (HITL) approach to accelerate workflows while preserving full human oversight and control. Data integrity and an immutable audit trail are secured via a two-layer distributed storage model combining IPFS and Hyperledger Fabric, which simultaneously serving as a blockchain-based Retrieval-Augmented Generation (RAG) knowledge base. The proposed architecture applies the Zero Trust threat model by eliminating implicit trust between components. Consequently, the AI module verifies the integrity and provenance of all input data via the blockchain before analysis.

Keywords: Cyber Security, Vulnerability Management, Blockchain, Hyperledger Fabric, IPFS, Artificial Intelligence, LLM, RAG, Zero Trust

1 Introduction

The complexity of software and hardware environments forming the foundation of the information society, along with the network devices connecting them, has increased exponentially over the past decade. In this heterogeneous digital ecosystem, the number of vulnerabilities stemming from configuration errors and implementation oversights is continuously rising [1]. Parallel to this, the toolkit available to attackers has evolved significantly, adversaries have begun weaponizing LLMs to automate phishing campaigns, generate polymorphic malware, and produce exploit scripts, with dark-web services such as WormGPT and FraudGPT making advanced attack capabilities accessible without technical expertise [2]. Consequently, dedicated security teams such as Security Operation Centers (SOCs) are no longer an optional safeguard but a fundamental operational requirement for organisations today. These teams are typically supported by technologies including Security Information and Event Management (SIEM) and Intrusion Detection and Prevention Systems (IDS/IPS) [3]. As the number of integrated components grows, so does the attack surface, where a single compromised element can undermine the integrity of the entire security. This motivates the adoption of a Zero Trust threat model, where no component implicitly trusts another.

To address the operational overload and scaling gap inherent in manual vulnerability processing, the integration of LLM-based AI into security workflows has emerged as a promising direction. LLMs represent a qualitative step beyond these limitations, they understand semantic context, enabling them to identify correlations between findings, prioritise vulnerabilities based on deployment-specific relevance, summarise complex reports, assign structured risk scores using established frameworks, and generate immediate, concrete remediation guidance.

However, deploying LLMs in security-critical workflows introduces reliability concerns. LLMs are prone to confabulation and context misinterpretation, in which superficially similar findings are incorrectly assessed. In a vulnerability management context, these failure modes carry direct operational consequences. For this reason, the LLM is treated as a decision-support component, meaning that a human analyst retains final authority over all assessments and remediation decisions. This design inherently satisfies the HITL principle recommended by the NIST AI Risk Management Framework [4].

This paper makes three contributions to the field of AI-assisted vulnerability management. First, it proposes a blockchain-based RAG architecture that guarantees the integrity and provenance of all LLM inputs via IPFS content-addressing and Hyperledger Fabric ledger entries, effectively mitigating PoisonedRAG-class attacks [10]. Second, it introduces a multi-model cross-validation strategy employing GPT-4 and Gemini 2.5 Pro in parallel to produce structured CVSS scores, STRIDE categories, and remediation suggestions, with divergences escalated to human review. Third, it presents a security analysis under a malicious-but-cautious adversary model, mapped to classical and AI-specific requirements.

2 Related Work

Academic literature confirms that vulnerability management is among the most resource-intensive processes in modern cybersecurity, frequently resulting in critical vulnerabilities remaining undetected due to alert fatigue [5]. Prior work has established that pairing AI-driven analysis with blockchain-enforced logging is a viable strategy for reducing false-positive noise in automated security pipelines. Nath et al. [6] proposed an AI and blockchain-based architecture for secure source code vulnerability detection, utilising a deep learning model for automated scanning alongside IPFS for off-chain storage and a permissioned blockchain for immutable audit logging. This work demonstrates the practical feasibility of decentralised, AI-assisted vulnerability management, yet does not address the challenge of processing the heterogeneous, high-volume output of general-purpose vulnerability scanners, nor the integrity of data fed to the AI models themselves. The frontier in this progression is the application of LLMs to vulnerability analysis. Sanvito et al. [7] investigate the capacity of LLMs to automate CVSS vector computation, finding that while current models demonstrate promising accuracy, they remain prone to scoring inconsistencies, directly motivating the multi-model collaboration strategy adopted in this work. The critical distinction of the present approach lies in the integrity of the retrieval layer. Unlike standard RAG systems, which rely on conventional vector databases with no authenticity guarantees, this approach uses a blockchain-recorded CID index. This ensures that every document retrieved by the AI model is verifiably unaltered, effectively eliminating the knowledge base as an attack surface for data poisoning. Yao et al. [8] catalogue the primary threat vectors, such as hallucination and data poisoning while the NIST AI Risk Management Framework explicitly identifies human oversight as a foundational control, providing regulatory-level justification for the HITL principle applied in the proposed architecture.

3 Proposed Architecture

The system integrates OWASP ZAP, Nessus, and SQLMap as primary data sources, targeting web application, infrastructure, and database vulnerability domains, respectively. However, storing large-scale reports directly on a blockchain is technically inefficient due to prohibitive storage costs. To address this, the architecture employs a two-layer distributed storage model, full reports are stored in IPFS, which assigns each document a unique Content Identifier (CID) computed as a cryptographic hash, while these CIDs are recorded on the Hyperledger Fabric ledger [9]. This structure creates an immutable, auditable index that serves as the foundation for a blockchain-based Retrieval-Augmented Generation (RAG) knowledge base, ensuring the LLM retrieves context exclusively from trusted sources. Ensuring the authenticity and integrity of data ingested by AI-driven analysis is important, as reports are exposed to data poisoning and RAG-targeted attacks [10].

During a vulnerability query, the AI module fetches the document from IPFS using the blockchain-recorded CID and verifies its integrity against the ledger record prior to inference. While the internal reasoning of the LLM remains a black box, the authenticity of every input is

cryptographically guaranteed, ensuring analytical outputs are grounded in verified, tamper-proof data. This design mitigates PoisonedRAG-class attacks, any unauthorized modification produces a different CID that does not match the ledger entry, making tampering detectable.

To address AI hallucination risks, the system operates as a Decision Support System (DSS). Both GPT-4 and Gemini 2.5 Pro are invoked in parallel against each verified record to produce structured JSON outputs, including CVSS scores, STRIDE categories, and prioritized remediation steps. Research indicates that cross-verification between independent models yields more reliable outputs, as divergences signal uncertainty [11]. When models produce divergent assessments, both results are presented to the analyst alongside a synthesized summary.

The central novelty of this architecture lies in the simultaneous realization of a blockchain-based RAG layer for tamper-proof verification and a multi-model cross-validation strategy two complementary controls that address the primary failure modes of LLM-driven security analysis. The architecture design diagram is available online [12].

4 Secure by Design and Security Analysis

In this section, a short security analysis of the architecture is provided, based on both classical and AI-specific security requirements, following NIST SP 800-53 [13] and NIST AI 600-1 [14]. Beyond the fundamental security requirements of Confidentiality, Integrity, and Availability, we consider Authentication and Non-repudiation as essential requirements for preventing unauthorised access and ensuring accountability. These are extended by Data Provenance to address the unique vulnerabilities of AI-integrated systems.

The analysis assumes a malicious-but-cautious adversary controlling one or more components of the architecture. Against this attacker model, GPT-4 and Gemini 2.5 Pro cross-validation with analyst escalation ensures that final human authority is maintained over all remediation actions, countering "cautious" manipulations that would otherwise go undetected.

The following defense measures are mapped to the stated requirements. Confidentiality is maintained through Fabric's private channels and end-to-end encryption, restricting data access to authorised participants even under partial network compromise. Integrity and Data Provenance are structurally enforced by IPFS content-addressing, where each report's CID is its cryptographic hash, making silent tampering detectable by construction. CIDs recorded on the Fabric ledger enable provenance verification before AI inference, effectively mitigating data poisoning and PoisonedRAG-type attacks. Authentication and Non-repudiation are guaranteed by the Membership Service Provider (MSP) through role-based access control and digitally signed transactions, preventing impersonation of legitimate nodes. Availability is ensured by the distributed architecture of both IPFS and Fabric, which eliminates single points of failure. Finally, Output Reliability is achieved by verifying input integrity before inference and storing AI evaluation results with their input CIDs on-chain, enabling post-hoc auditability and reproducibility.

5 Conclusion

This paper presented a security architecture integrating automated vulnerability scanning, LLM-driven analysis, and a blockchain-based RAG knowledge base. IPFS-derived CIDs recorded on the Hyperledger Fabric ledger enforce the Zero Trust threat model, ensuring cryptographically verified provenance for all LLM inputs. A multi-model cross-validation strategy reduces hallucination and scoring inconsistency, while the HITL principle preserves analyst authority over all remediation decisions.

References

[1] CVE Details. Browse CVE Vulnerabilities by Date, *CVE Details - The Ultimate Security Vulnerability Datasource*, 2024.

[2] S. Mutiuddin, O. Ibrahim, K. Rayan. Weaponizing Large Language Models: Automated Phishing, Social Engineering, and Malware Generation, *International Journal of Engineering Technology Research and Management*, vol. 10, no. 1, pp. 33–38, 2026.

[3] W. S. Admassa, Y. Munaye, A. Diro. Cyber security: State of the art, challenges and future directions, *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 49–61, 2024.

[4] NIST. Artificial Intelligence Risk Management Framework, *NIST AI 100-1*, National Institute of Standards and Technology, Gaithersburg, 2023.

[5] ISC2. *Cybersecurity Professionals Navigate Evolving Workplaces While Seizing New Opportunities*, 2025.

[6] P. Nath et al. AI and Blockchain-based source code vulnerability detection and prevention system for multiparty software development, *Computers and Electrical Engineering*, vol. 106, 2023.

[7] D. Sanvito, G. Arriciati, G. Siracusano, R. Bifulco, M. Carminati. AutoCVSS: Assessing the Performance of LLMs for Automated Software Vulnerability Scoring, *Proceedings of EMNLP 2025 Industry Track*, pp. 564–575, 2025.

[8] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, Y. Zhang. A Survey on Large Language Model (LLM) Security and Privacy: The Good, The Bad, and The Ugly, *High-Confidence Computing*, vol. 4, no. 2, p. 100211, 2024.

[9] I. Bashir. Mastering Blockchain: Inner workings of blockchain, from cryptography to DeFi, NFTs and Web3, *Packt Publishing*, Fourth edition, 2023.

[10] W. Zou, R. Geng, B. Wang, J. Jia. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models, *Proceedings of the 34th USENIX Security Symposium*, 2025.

[11] A. Margavi, I. Jebellat, E. Jebellat, S. Davoudi. Enhancing Answer Reliability Through Inter-Model Consensus of Large Language Models, *IFIP Advances in Information and Communication Technology*, vol. 758, Springer, Cham, 2025.

[12] C. Nagy. Architecture Design Diagram for Blockchain-Based RAG Vulnerability Management, *GitHub repository*, [Online]. Available: https://github.com/ncsn/Blockchain-based-RAG-Architecture-for-Vulnerability-Assessment, 2026.

[13] NIST. Security and Privacy Controls for Information Systems and Organizations, *NIST Special Publication 800-53, Revision 5*, National Institute of Standards and Technology, 2020.

[14] NIST. Artificial Intelligence Risk Management Framework: Generative AI Profile, *NIST AI 600-1*, National Institute of Standards and Technology, 2024. Available:

Towards Automated Generation of Adversarial Malware Samples with a Genetic Algorithm

József Sándor, Bence Kovács, and Levente Buttyán

Abstract: Machine-learning-based malware detection poses a particular challenge for IoT devices with limited resources. SIMBIoTA-ML has proven effective at identifying variants of known IoT malware families, but its robustness against evasion attacks is insufficient. This paper presents GAME (Genetic Algorithm for Malware Evasion), a framework that automatically generates adversarially crafted malware samples using a Genetic Algorithm (GA). Instead of patching ELF executables at the byte level, GAME introduces format-preserving modification strategies whose parameters are evolved by the GA to find optimal evasion settings for a given binary. GAME reliably generates evasion strategies whose adversarial examples successfully deceive SIMBIoTA-ML, making them well-suited for adversarial training to make the detector more robust.

Keywords: IoT Malware Detection, Adversarial Examples, Genetic Algorithm

1 Introduction

The proliferation of Internet of Things (IoT) devices has dramatically expanded the cybersecurity attack surface. Unlike traditional IT infrastructure, IoT environments are characterized by heterogeneous architectures and severe resource constraints, making them attractive targets for large-scale botnet operations. The devastating impact of this paradigm shift was demonstrated by the Mirai botnet and sustained by subsequent polymorphic families such as Qbot and SpyBot.

The industry has moved beyond rigid signature-based detection toward similarity-based and machine learning (ML) approaches. SIMBIoTA-ML [2] is a prime example: an effective ML-based malware detector for IoT devices with lightweight resource requirements. However, new detection techniques are not necessarily robust against deliberately crafted adversarial examples (AEs), i.e., modified malware samples that evade detetcion by a specific anti-malware system while preserving full malicious functionality.

Adversarial training is an approach to make ML-based malware detectors more robust against evasion attacks by using AEs too in their training. This requires high-volume, diverse, and valid AE datasets. The central problem addressed in this paper is the design of an automated framework capable of producing such datasets without human intervention. Our approach is to use a Genetic Algorithm (GA) to traverse the solution space, where the evolutionary process is grounded in rigorously defined modification primitives – atomic, format-preserving operations that allow the algorithm to alter a binary's structural and statistical footprint without disrupting execution flow. Our framework, called GAME, reliably generates AE creation strategies, which produce AEs that deceive SIMBIoTA-ML. Hence, ultimately, GAME can be used in the adversarial training of ML-based detectors to make them more robust.

2 Background

SIMBIoTA-ML [2] is a lightweight machine learning-based malware detector for embedded IoT devices. It trains a Random Forest classifier using feature vectors extracted from TLSH hashes [1]. SIMBIoTA-ML reports a malware detection rate of approximately 95% while maintaining a low false positive rate. Prior adversarial strategies against SIMBIoTA-ML include the Chunker (appending chunks of the malware to itself to distort its statistical profile) and Disguiser (appending a benign file to drag the feature vector toward the benign cluster) [3]. While adversarial

training against these known strategies improves robustness, it inherently relies on the defender anticipating the attacker's logic. This paper argues that fixed strategies are insufficient proxies for determined attackers and proposes a GA to discover novel, unforeseen evasion techniques.

The **Executable and Linkable Format (ELF)** is the standard binary format for Linux-based systems, covering the vast majority of IoT firmware running on ARM, MIPS, and PowerPC architectures. The malware samples targeted in this work are ELF binaries as well. To evade SIMBIoTA-ML detection, these binaries must be modified in a way that alters their TLSH hash values, while preserving full execution functionality – a non-trivial constraint, as malformed binaries are immediately rejected by the operating system's loader. To ensure all generated AEs remain valid and functional, GAME uses the LIEF (Library to Instrument Executable Formats) library, which handles the low-level structural bookkeeping of ELF manipulation automatically.

Genetic Algorithms are stochastic search heuristics inspired by natural selection, designed for optimization over vast, discrete, non-differentiable search spaces [4]. In this work, we employ a GA in which the population consists of modification strategies (genotypes) rather than raw binary files [5]. Each strategy is evaluated by instantiating it as a set of concrete modified binaries (phenotypes). The evolutionary operators (mutation, crossover) act upon the genotypes, and the fitness evaluation is performed on the phenotypes.

3 Approach and Implementation

The framework operates as a closed-loop evolutionary system. Given a functional, malicious ELF binary as input, it iteratively evolves a population of modification strategies to evade the detection of SIMBIoTA-ML. The system is black-box: it requires no knowledge of the detector's internal state, relies solely on the prediction label. Each generational cycle consists of the following phases:

1. Population Creation: N strategies are initialized randomly in the first generation to ensure broad search space coverage.
2. Phenotype Evaluation: Each strategy is executed multiple times; each run produces a distinct modified ELF binary. The strategy's fitness is the average fitness across its phenotypes, ensuring statistical robustness.
3. Selection: Tournament selection with elitism preserves the top-k strategies unchanged.
4. Crossover and Mutation: Selected parents are recombined and subjected to mutation – structural changes (swapping injection primitives or data sources) and parameter-level tuning.
5. Re-population: The next generation is composed of elites, fresh random strategies, and offspring.

The fundamental unit of evolution is the Gene, representing one atomic modification. A complete strategy is defined as an ordered sequence of these genes, allowing for potential chaining of modifications. Formally, each gene G is defined as:

$$G = \langle f_{slot}, f_{data}, \mathbf{P}_{dist}, \mathbf{P}_{meta} \rangle \tag{1}$$

where f_{slot} selects the injection primitive (append, padding overwrite, or segment injection), f_{data} selects the payload data generation function, $\mathbf{P}_{dist}$ encodes the statistical parameters of the data generator (e.g., mean), and $\mathbf{P}_{meta}$ encodes global constraints (e.g., payload size).

Three format-preserving **modification primitives** (f_{slot}) are implemented using the `LIEF` library:

- Data Appending: Appends a generated payload to the physical end of the ELF file. Requires no structural parsing; highly effective but increases file size.
- Padding Overwriting: Exploits memory alignment gaps between ELF segments by overwriting unused padding bytes with generated data, incurring zero file size overhead.
- Segment Injection: Instantiates a new `PT_LOAD` segment and relies on `LIEF`'s builder to automatically recalculate virtual addresses and header offsets, integrating the payload into the binary's memory map.

To defeat the hash calculation mechanism of the TLSH, injected payloads must exhibit a statistical texture similar to legitimate binaries. Two categories of **data generation** (f_{data}) are implemented:

- Distribution-Based Generation: Uses Inverse Transform Sampling over Normal, Gamma, and Weibull distributions to synthesize bytes whose frequency profiles mimic machine code or compressed data regions. Distribution parameters are evolved by the GA.
- File-Based Generation: Transplants byte sequences from benign ELF binaries using sequential sampling (preserving local structure) or random block sampling (preserving the global histogram while destroying local patterns).

The **fitness function** balances evasion effectiveness against stealth, thus modeling the attacker's objectives. The primary driver is the TLSH difference score, reflecting the core evasion goal. Secondary stealth constraints include size overhead and entropy deviation. Stealth constraints penalize naive strategies, biasing the algorithm toward discovering sophisticated techniques. Critically, if the detector classifies the phenotype as malicious, the fitness score is harshly reduced, making detection evasion the dominant objective.

4 Experiments and Results

For the experiments, we used ARM binaries: 2,000 malware samples[1] and 2,000 benign files extracted from firmware images. These samples served as the training data for SIMBIoTA-ML. After training, we applied GAME to a subset of the malware samples to generate evasion strategies against the detector.

The experimental results show a consistent and rapid convergence pattern (see Figure 1). The initial random population (first generation) typically exhibited low fitness, with most individuals falling below the detection threshold. Within 5 to 10 generations, the average population fitness consistently spiked. By generation 32, over 90% of evolved samples successfully bypassed SIMBIoTA-ML.

Injecting a small number of fresh random strategies per generation consistently produced earlier breakthroughs in detection evasion, particularly for samples that proved difficult to evade. Without fresh strategies, the population could stagnate for many generations without crossing the detection threshold. Mutation rate analysis revealed a trade-off: low mutation rates sufficed when the initial population already contained strong candidates, enabling stable hill-climbing convergence. Higher mutation rates were necessary for weaker initial populations to ensure adequate exploration, but increased the risk of losing good strategies. This trade-off is managed by the elitism mechanism, which guarantees that the best solutions are never discarded.

Padding Overwrite and Segment Injection consistently dominated the top fitness tier. Since padding overwrite adds zero file size overhead, it achieves near-perfect scores on the stealth components of the fitness function. Segment injection proved similarly robust. The Append Data

[1] `https://github.com/CrySyS/cube-maliot-2021`

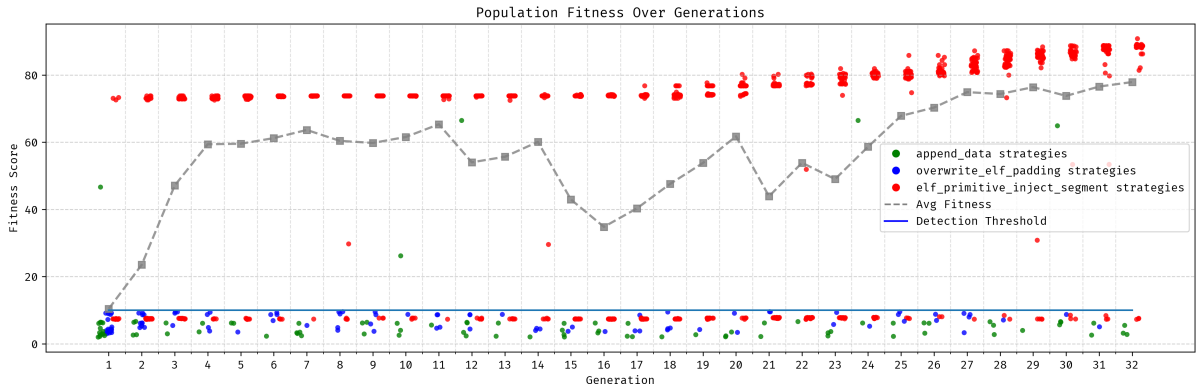


Figure 1: Evolution of strategies for creating AEs in case of a Mirai malware variant.

strategy is effective at evasion but penalized by size overhead. The evolution of data distribution parameters resulted in payloads closely matching the entropy of the original code, effectively masking modifications from entropy-based heuristics. Successful AEs typically exhibited a file size increase of less than 5-10%, significantly lower than naive appending approaches.

5 Conclusion

This paper presented an automated evolutionary framework for generating AEs targeting ML-based IoT malware detectors, such as SIMBIoTA-ML. By defining a semantic genetic representation of modification strategies and implementing format-preserving binary manipulation primitives, the system reliably evolves evasion strategies without human intervention. Our experiments confirmed that the adversarial search space for ELF binaries is vast, with the GA consistently achieving over 90% evasion rates within 32 generations. Stealth-oriented primitives – particularly padding overwrite – proved dominant due to their zero file size overhead. The primary defensive application of this work, also our future work, is adversarial training: integrating the GAME framework into the training loop of malware detectors to expose their model to novel AEs, thus increasing their robustness.

References

[1] J. Oliver, C. Cheng, and Y. Chen, "TLSH - A Locality Sensitive Hash," in Proc. 4th Cybercrime and Trustworthy Computing Workshop, IEEE, 2013.

[2] D. Papp, G. Acs, R. Nagy, and L. Buttyan, "SIMBIoTA-ML: Light-Weight, Machine Learning-Based Malware Detection for Embedded IoT Devices," in Proc. IoTBDS, 2022.

[3] J. Sandor, R. Nagy, and L. Buttyan, "Increasing the Robustness of a Machine Learning-Based IoT Malware Detection Method with Adversarial Training," in Proc. ACM Workshop on Wireless Security and Machine Learning, 2023.

[4] J. R. Koza, "Genetic Programming as a Means for Programming Computers by Natural Selection," Statistics and Computing, 1994.

[5] F. Wang et al., "Binary Black-Box Adversarial Attacks with Evolutionary Learning against IoT Malware Detection," Wireless Commun. Mobile Comput., 2021.

Keystroke-Based Behavioral Liveness Detection for Adaptive Multi-Factor Authentication

Yusufbek Sulaymonov, Dr.Veronika Szücs

Abstract: The increasing sophistication of automated interaction and identity fraud poses fundamental challenges for authentication in future computing environments. While keystroke dynamics has been widely studied for closed-set user identification, its application to open-set behavioral liveness detection—determining whether input originates from a live human rather than synthetic or replayed sources—remains insufficiently explored. This study formulates keystroke-based liveness detection as an anomaly detection problem under realistic adversarial deployment conditions. The model is trained exclusively on legitimate human data, without assuming access to attacker-specific samples. A two-phase experimental framework is introduced: a large-scale analysis of a public dataset comprising over 136 million keystrokes from approximately 168,000 real users, complemented by controlled data collection and synthetic input generation simulating machine-generated and replayed typing patterns. Supervised and unsupervised approaches are evaluated under explicit open-set conditions. Results demonstrate strong separability between genuine and synthetic input (AUC $= 0.95$), achieving a False Positive Rate of approximately 0.2% in the optimized configuration. Higher-order temporal dependencies and behavioral variability emerge as stable discriminative features for distinguishing structurally implausible interaction patterns. These findings support the feasibility of keystroke-based behavioral liveness detection as a scalable and unobtrusive signal that can be integrated into adaptive multi-factor authentication architectures.

Keywords: keystroke dynamics; behavioral liveness detection; open-set recognition; multi-factor authentication; adaptive authentication

1 Introduction

Future computing systems increasingly operate in distributed, large-scale, and automation-intensive environments, where interactions between humans and automated agents occur seamlessly at shared interfaces. As credential abuse, scripted interaction, and AI-driven automation become more sophisticated, authentication mechanisms based solely on static identifiers or discrete login events are becoming insufficient. In such contexts, ensuring that ongoing interaction originates from a genuine human user rather than from an automated process represents not merely a security feature, but an architectural requirement for resilient digital systems.

Behavioral biometrics offer a promising direction for addressing this challenge. Among these modalities, keystroke dynamics has been extensively studied as a means of user identification and verification. Prior research has predominantly focused on closed-set scenarios, where the objective is to distinguish among known individuals based on relatively stable typing profiles. While such approaches have demonstrated high classification accuracy, they are typically not designed to operate under open-set, adversarial conditions in which non-human input strategies may evolve, remain partially unknown, or intentionally mimic human statistical properties.

This study addresses the following research problem: can keystroke dynamics serve as a reliable interaction-level liveness signal in large-scale computing environments where automated agents attempt to replicate human typing behavior? Rather than treating keystroke dynamics as a mechanism for identifying specific individuals, we investigate its suitability as a behavioral plausibility indicator capable of distinguishing genuine human interaction from synthetic or scripted input. This perspective shifts the emphasis from identity recognition toward system-level robustness and continuous-authentication-oriented interaction assessment under adversarial uncertainty.

To operationalize this problem, the study addresses three interrelated research questions. First, to what extent can keystroke dynamics provide a reliable behavioral signal for discriminating between genuine human interaction and machine-generated typing under open-set adversarial conditions (RQ1)? Second, how do supervised classification and anomaly detection paradigms compare in terms of robustness and generalization when adversarial input distributions are incomplete or only partially observable during training (RQ2)? Third, does keystroke-based open-set liveness detection constitute a structurally suitable behavioral signal for background continuous authentication within adaptive multi-factor authentication architectures (RQ3)?

The primary conceptual contribution of this work is a reframing of keystroke biometrics as an open-set liveness detection problem under adversarial deployment conditions. Traditional keystroke-based studies predominantly address closed-set user identification, where all relevant classes are assumed to be known during training. In contrast, this work models only legitimate human behavior and treats non-human input as an undefined, potentially evolving class. By explicitly integrating threat modeling, open-set recognition principles, and system-level deployment considerations, the study shifts the focus from identity classification accuracy to behavioral plausibility and robustness against synthetic interaction.

2 Background and Related Work

Keystroke dynamics has been extensively investigated as a behavioral biometric modality for user identification and authentication. Early studies demonstrated that timing-related features—such as key hold durations and inter-key latencies—exhibit user-specific characteristics that can be modeled for classification and verification purposes [1, 2]. Subsequent research expanded this paradigm toward continuous authentication, where identity is assessed implicitly during ongoing interaction rather than through isolated login events [4, 5]. Behavioral variability and long-term stability of typing patterns have been examined in detail [3], highlighting both the persistence of user-specific traits and the natural fluctuations introduced by cognitive load, fatigue, and contextual factors.

Despite this progress, the majority of existing work remains focused on distinguishing among enrolled users or improving classification accuracy under relatively controlled assumptions. Comparatively fewer studies address the system-level challenge of liveness detection, particularly the discrimination between genuine human input and machine-generated typing behavior intentionally designed to mimic human interaction patterns. Recent surveys explicitly identify liveness assessment and adversarial robustness as open research challenges in behavioral biometrics [6].

From a methodological perspective, most keystroke-based systems operate under closed-set classification assumptions. The human–bot discrimination problem inherently exhibits open-set characteristics, where adversarial interaction patterns may not be fully observable during training. Open-set recognition has been studied extensively in pattern recognition and computer vision [7, 8], emphasizing the need to model unknown classes and reject unfamiliar inputs. Despite these advances, the integration of open-set modeling principles into behavioral biometric liveness detection remains limited.

Recent studies have begun to address synthetic behavioral input more directly. Gonzales et al. [9] introduced tools and datasets for generating human-like synthetic keystroke sequences to evaluate the robustness of detection models. Their findings indicate that simple statistical similarity is insufficient for reliable discrimination, underscoring the importance of capturing higher-order temporal structure. Research on large-scale automated interaction and bot detection similarly demonstrates that temporal input features can provide valuable signals for distinguishing human and non-human activity [10, 11]. However, such approaches are rarely embedded within authentication architectures as a dedicated liveness layer.

Multi-factor authentication (MFA) frameworks increasingly combine knowledge-based, possession-based, and biometric factors to improve security [12, 13]. Behavioral biometrics have been proposed as a passive complementary factor capable of enhancing security while preserving usability. Nevertheless, the integration of keystroke-based liveness detection into adaptive and scalable MFA architectures remains underexplored, particularly under open-set and adversarial assumptions.

3 Materials and Methodology

3.1 Liveness Detection and Behavioral Biometrics

In biometric authentication systems, liveness detection mechanisms complement identity verification by assessing whether the interaction itself is plausibly human. Existing strategies range along a continuum of user participation: passive approaches infer human presence implicitly from natural behavioral signals; active approaches rely on explicit user challenges; hybrid strategies combine both to balance robustness and usability. Within this landscape, keystroke dynamics is well suited for passive and hybrid liveness detection because its temporal interaction signals emerge naturally during system use.

3.2 Keystroke Feature Representation

Typing behavior can be decomposed into temporal and behavioral features that describe interaction dynamics rather than the semantic content of the typed text. Table 1 provides formal definitions of the feature set used in this work.

Table 1: Keystroke dynamics features used for representation and modeling.

Feature	Definition
Dwell time H_k	$H_k = t_k^{\uparrow} - t_k^{\downarrow}$: time a key remains pressed.
Press latency $P_{i\to j}$	$P_{i\to j} = t_j^{\downarrow} - t_i^{\downarrow}$: time between consecutive key-down events.
Release latency $R_{i\to j}$	$R_{i\to j} = t_j^{\uparrow} - t_i^{\uparrow}$: time between consecutive key-up events.
Inter-key latency $F_{i\to j}$	$F_{i\to j} = t_j^{\downarrow} - t_i^{\uparrow}$: release-of-previous to press-of-next (flight time).
Total duration T_{seq}	$T_{\text{seq}} = t_{\text{last}}^{\uparrow} - t_{\text{first}}^{\downarrow}$: full session span.
Key distance $D_{i,j}$	Euclidean distance between key centers on the keyboard layout.

The primary features are dwell time, press latency, release latency, and inter-key latency, complemented by session-level characteristics such as typing speed, error rate, correction strategy, pause-burst patterns, and Shift key overlaps. These provide behavioral context reflecting interaction irregularities that may distinguish human from automated behavior. Prior studies suggest that more subtle indicators—such as response time to locate specific keys or the influence of key-pair distance on typing fluidity—can further strengthen discriminative power [15].

3.3 Threat Model

The primary adversarial objective is to inject machine-generated or automated input into an interface while being classified as genuine human interaction. The attacker is assumed to have knowledge of general population-level typing statistics and may generate synthetic input

by sampling from empirical timing distributions, but does not have access to internal model parameters or exact decision thresholds—a realistic gray-box scenario.

Three categories of adversarial behavior are considered: (i) rigid automated typing with fixed or near-constant timing intervals; (ii) statistically human-like synthetic typing generated through distribution-based sampling; and (iii) non-sequential text insertion such as clipboard pasting. Paste-based input differs fundamentally from sequential synthetic typing: it bypasses intermediate keystroke events entirely and lacks the temporal micro-structure characteristic of genuine human interaction. In practical deployment, paste events can be identified through event-level inconsistencies, so the empirical evaluation focuses primarily on sequential synthetic typing, which constitutes the more challenging and structurally ambiguous adversarial class.

3.4 Classification and Anomaly Detection Framework

Human–machine discrimination can be operationalized using either supervised classification or anomaly detection. Supervised classification assumes labeled examples of both human and non-human input and learns explicit decision boundaries. In anomaly detection, the model learns the distribution of legitimate human behavior and treats deviations as potential indicators of automation. This paradigm is particularly relevant when machine-generated input is scarce, evolving, or incomplete—conditions that are realistic in open-set deployment. In this study, both paradigms are evaluated: Random Forest classifiers for supervised experiments, and Isolation Forest and One-Class SVM (OC-SVM) for anomaly detection.

4 Experimental Setup

Figure 1 presents the complete experimental workflow, covering data acquisition, preprocessing, feature extraction, modeling, and evaluation.

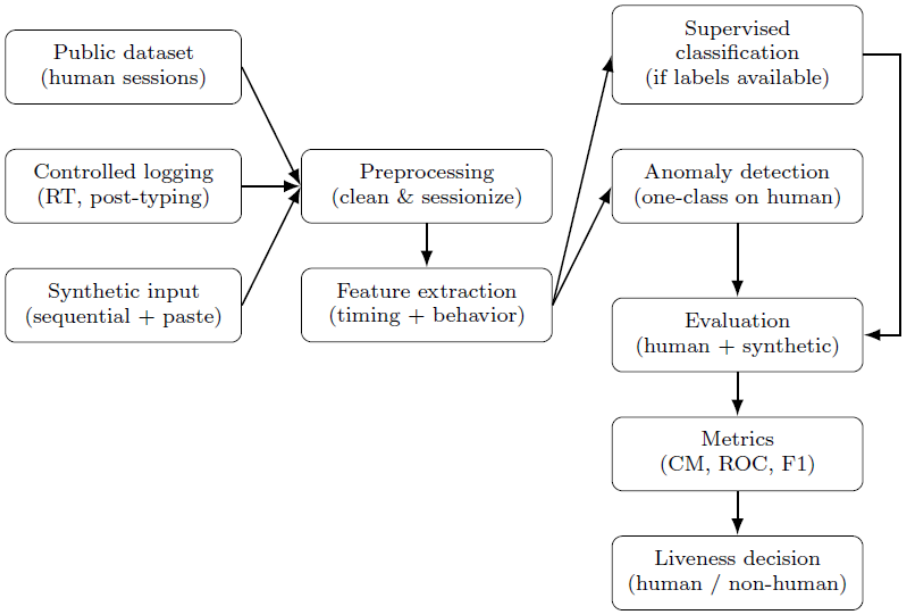


Figure 1: Overview of the experimental pipeline for keystroke-based liveness detection.

4.1 Data Sources

Public human keystroke dataset. The primary source of genuine human typing data was a large-scale publicly available dataset containing approximately 136 million keystrokes from more than 168,000 users [14]. Each record contains a unique participant identifier (`participant_id`), a unique typing session identifier (`test_section_id`), the reference (sample) text, the user-typed text, the ASCII code of the pressed key, and precise timestamps for key press and release events. These fields allow detailed session-level analysis and the computation of derived timing features without storing raw biometric patterns. User identifiers were used solely for session grouping and were not used as classification targets; liveness detection operates independently of user identity. From this dataset, 14,156 genuine human sessions were extracted for model training, drawn from the full heterogeneous population to ensure representative coverage of real-world typing variability.

Synthetic machine-generated input. Since publicly available corpora of real automated typing are scarce, synthetic input sequences were generated using a Python-based simulation framework parameterized by empirical statistics derived from the public dataset. Two categories were produced. The first represents rigid automated typing with near-constant timing intervals. The second simulates statistically human-like input by sampling dwell times and inter-key intervals from Gaussian distributions ($\mu_{\text{hold}} = 116.24\,\text{ms}$, $\sigma_{\text{hold}} = 23.88\,\text{ms}$; $\mu_{\text{IKI}} = 238.66\,\text{ms}$, $\sigma_{\text{IKI}} = 111.6\,\text{ms}$) with a lower bound of 60 ms enforced to prevent physically implausible artifacts. A third category—clipboard paste insertion—was also considered but treated separately, as it lacks sequential keystroke structure and is detectable through event-level inconsistencies. The evaluation therefore focuses on sequential synthetic typing, which constitutes the more structurally ambiguous adversarial class.

An important limitation is that only distribution-based generators were used in this study. More advanced AI-based generators—such as LSTM or GAN architectures trained to produce sequential typing patterns—represent a stronger adversarial threat and were not evaluated here. This is explicitly acknowledged as a direction for future work; current results should be interpreted as a lower bound on adversarial robustness.

Controlled human data collection. Additional data were collected using a custom Python-based logging application that recorded timestamps for stimulus presentation, key press and release events, and submission actions. This enabled extraction of pre- and post-typing behavioral indicators—reaction time (`first_key_time − sentence_display_time`) and post-typing submission delay—that extend beyond conventional dwell and flight metrics and are absent from the public dataset.

4.2 Feature Extraction and Aggregation

Raw keystroke event streams were transformed into session-level feature vectors. Sessions were defined using the `participant_id` and `test_section_id` identifiers from the public dataset; where these were unavailable, artificial sessions were constructed by grouping consecutive keystrokes into fixed windows of 50 keystrokes. Event-level timing variables (dwell time, press latency, release latency, inter-key latency) were aggregated using mean, standard deviation, and relative variability (coefficient of variation) per session. Behavioral indicators—including pause-burst count (inter-key latency $> 500\,\text{ms}$), WPM typing speed (standardized: 5 characters per word), correction count (Backspace and Delete presses), Shift key overlap occurrences, and textual error rate via Levenshtein distance—were also extracted to capture interaction irregularities beyond basic timing.

4.3 Model Configuration and Evaluation Protocol

For supervised experiments, labeled datasets were constructed with binary labels (human or synthetic) and partitioned 80/20 for training and testing.

For anomaly detection, models were trained exclusively on the 14,156 genuine human sessions described above, with no synthetic samples used during training. Evaluation was conducted on a combined test set of equal size containing both human sessions and synthetically generated machine input. Isolation Forest and OC-SVM were initialized with default parameters as baselines, followed by systematic sensitivity analysis over $\nu \in \{0.01, 0.05, 0.08, 0.09, 0.1, 0.15\}$ and $\gamma \in \{\text{auto}, 0.1, 0.01\}$. Parameter selection was guided by the threat model, prioritizing minimization of false acceptance (machine input classified as human) over aggregate accuracy.

5 Results

5.1 Supervised Classification: Random Forest

As a baseline comparison, a Random Forest classifier was trained on a labeled dataset containing both human and synthetic sessions using an 80/20 train/test split. The most influential feature in the trained model was `press_latency_std_to_mean` (the ratio of standard deviation to mean press latency), reflecting high variability between key presses as a characteristic human signal. Other prominent features included dwell time variability and inter-key latency dispersion.

Although test-set accuracy appeared near-perfect, the classifier exhibited a critical generalization failure: it overfitted to the specific synthetic patterns used during training and could not reliably distinguish human input from unseen or structurally different machine-generated patterns. This behavior—high apparent accuracy alongside poor generalization—stems directly from the asymmetry between the rich, heterogeneous human data and the limited coverage of synthetic patterns. In particular, the classifier learned to recognize specific generator artifacts rather than general human behavioral structure.

These results motivate the shift to anomaly detection. By modeling only the distribution of legitimate human behavior and treating any deviation as a potential anomaly, the one-class paradigm avoids dependence on complete adversarial coverage and generalizes naturally to unseen attack patterns.

5.2 Baseline Anomaly Detection Performance

Table 2 reports performance metrics for the baseline OC-SVM configuration ($\nu = 0.05$). The model was trained exclusively on human sessions and evaluated on the combined test set. Strong overall separability is achieved across both classes.

Table 2: Baseline anomaly detection performance (OC-SVM, $\nu = 0.05$).

	Precision	**Recall**	**F1-score**	**Support**
Machine	0.93	0.95	0.94	14,156
Human	0.95	0.93	0.95	14,156

In liveness detection scenarios, aggregate metrics must be interpreted cautiously, as the security impact of false acceptances (machine input classified as human) differs fundamentally from that of false rejections. Under the baseline configuration, a non-negligible proportion of machine-generated inputs were incorrectly classified as human, representing the primary security risk under the defined threat model.

5.3 Confusion Matrix Analysis

Confusion matrices were examined for different parameter configurations to analyze error structure. From each matrix, the True Positive Rate (TPR) and True Negative Rate (TNR) were derived:

$$\text{TPR} = \frac{TP}{TP + FN}, \qquad \text{TNR} = \frac{TN}{TN + FP}. \tag{1}$$

Figure 2 shows the baseline confusion matrix. Figure 3 shows the optimized configuration.

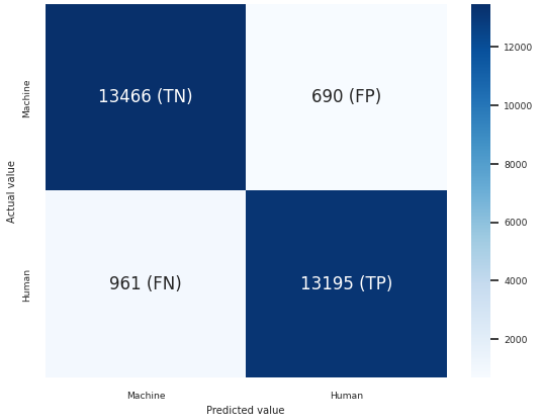


Figure 2: Confusion matrix: baseline OC-SVM ($\nu = 0.05$).

5.4 Optimized Model Performance

Hyperparameter tuning was performed by systematically varying the ν and γ parameters of the OC-SVM. Under the optimized configuration ($\nu = 0.1$), only 25 out of 14,156 machine-generated samples were misclassified as human input (Figure 3), corresponding to:

$$\text{FPR} \approx 0.18\%, \qquad \text{TNR} = 0.998.$$

The model simultaneously retained high sensitivity to genuine human input, maintaining a strong True Positive Rate. This reflects a conservative operating regime in which the decision boundary is adjusted to reduce adversarial acceptance while tolerating limited human rejection—consistent with deployment in background liveness monitoring or multi-factor authentication contexts.

5.5 ROC Analysis

Receiver Operating Characteristic (ROC) analysis evaluated threshold-dependent discrimination performance across varying decision thresholds. The optimized model achieved an AUC of 0.95 (Figure 4), indicating strong separability between genuine human interaction and synthetic machine-generated patterns. This result confirms that the learned behavioral boundary captures structural characteristics of human typing beyond simple statistical similarity, supporting the feasibility of anomaly-based liveness detection under open-set deployment conditions.

6 Discussion

6.1 Reflection on the Research Questions

RQ1. The empirical results provide strong evidence that keystroke dynamics can serve as a reliable behavioral liveness signal. The anomaly detection approach achieved robust separa-

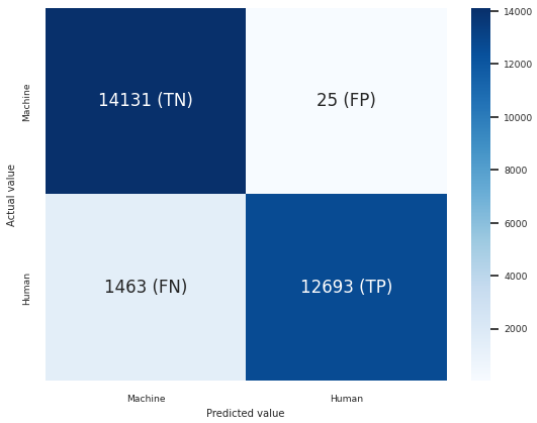


Figure 3: Confusion matrix: optimized OC-SVM ($\nu = 0.1$).

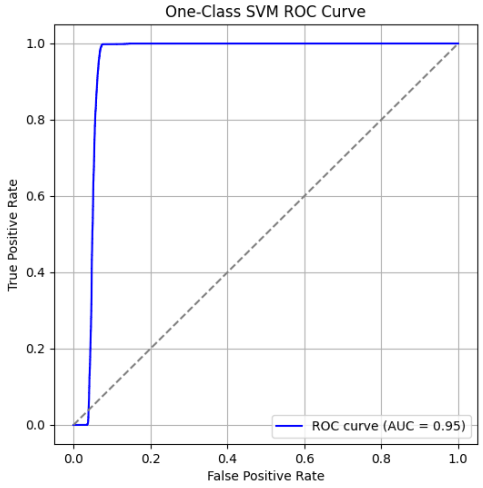


Figure 4: ROC curve for the optimized One-Class SVM model (AUC $= 0.95$).

tion between human and synthetic input, with an AUC of 0.95 and a False Positive Rate of approximately 0.2% in the optimized configuration. Session-level structural characteristics of typing behavior contain sufficient discriminative information to detect structurally implausible interaction patterns, even when synthetic input approximates population-level timing statistics. The results suggest that discrimination is not driven solely by absolute timing magnitudes, but by higher-order structural properties—variability ratios, pause distributions, and correction dynamics. While synthetic sequences may approximate mean dwell or flight times, they struggle to reproduce the micro-irregularities and dispersion patterns characteristic of natural human typing.

RQ2. The supervised Random Forest baseline illustrates the core problem directly: despite near-perfect test-set accuracy, the classifier failed to generalize to unseen machine patterns because it learned generator-specific artifacts rather than general human behavioral structure. This is a consequence of the asymmetry between rich human data and limited synthetic coverage—a condition that holds in any realistic open-set deployment. In contrast, anomaly detection models only the distribution of legitimate human behavior, making no assumptions about the form of non-human input. This design choice produces robustness to unseen and evolving attack strategies, at the cost of some sensitivity on the training distribution. The experimental results confirm this trade-off: the optimized OC-SVM achieves strong separation (AUC $= 0.95$) without

any adversarial training samples.

RQ3. The empirical findings demonstrate that the extracted feature space captures stable structural properties of human interaction suitable for background continuous authentication. Keystroke-based open-set liveness detection can serve as a passive behavioral signal within adaptive multi-factor authentication architectures, contributing to dynamic risk assessment without functioning as a standalone identity gate. Because it operates unobtrusively and relies on naturally occurring interaction data, it is particularly suitable for continuous risk assessment in automation-intensive environments. Occasional false rejections may be mitigated through graduated trust adjustment or step-up authentication mechanisms, thereby preserving usability while maintaining robustness.

6.2 Comparison with Existing Approaches

Table 1 summarizes key differences between representative approaches and the present work. Unlike closed-set classifiers that assume all relevant classes are known at training time, the anomaly detection framework models only legitimate human behavior and treats deviations as structurally implausible interaction. This design choice directly responds to limitations identified in open-set recognition research [7]. Furthermore, by incorporating an explicit threat model and evaluating performance against structured synthetic adversarial patterns, the study bridges behavioral biometrics with deployment-oriented robustness considerations.

Table 3: Comparison of representative keystroke-based approaches and the present work.

Study Type	**Set**	**Primary Goal**	**Adversarial Modeling**	**Continuous Use**
Classical identification	Closed	User recognition	Limited / none	Rare
Continuous authentication	Closed	Identity persistence	Limited	Yes
Bot detection	Mixed	Human vs. bot	Often statistical	Rarely in MFA
Present work	Open	Liveness detection	Explicit threat model	Background use

6.3 Limitations and Future Research

Several limitations warrant consideration. Machine-generated input was produced using distribution-based Gaussian sampling, which approximates population-level statistics but does not represent the full adversarial threat landscape. In particular, AI-based keystroke generators—such as LSTM sequence models or generative adversarial networks trained on human typing corpora—are capable of producing temporally coherent sequences that more closely reproduce higher-order dependencies and micro-variability characteristic of genuine human input. Testing robustness against such generators is a necessary direction for future work, and the current AUC of 0.95 should be interpreted as performance against distribution-based adversaries rather than the most sophisticated possible attack. Contextual factors such as device type, keyboard layout, and environmental variability were not fully controlled. Performance was evaluated at the session level rather than through streaming authentication pipelines.

Future work should investigate AI-based generative adversaries, cross-context generalization, streaming evaluation protocols, and hybrid anomaly–supervised frameworks. Longitudinal studies would assess behavioral stability and drift under extended deployment.

7 Conclusion

This study investigated keystroke dynamics as a behavioral signal for liveness detection under open-set and adversarial deployment assumptions. Rather than framing the problem as closed-set user identification, the work reformulated human–machine discrimination as an anomaly detection task in which only legitimate human behavior is modeled during training. This modeling paradigm directly addresses a key limitation of many biometric systems: the impracticality of exhaustively enumerating non-human or adaptive adversarial interaction strategies.

Empirical evaluation demonstrated that session-level structural characteristics of human typing—particularly variability patterns and higher-order temporal structure—provide sufficient discriminative information to separate genuine interaction from synthetic input. The optimized One-Class SVM configuration achieved strong separability (AUC $= 0.95$) while maintaining a low false acceptance rate (FPR $\approx 0.2\%$) without any adversarial training samples.

As a passive and unobtrusive behavioral signal, keystroke-based liveness detection can be integrated into adaptive multi-factor authentication architectures and continuous-authentication-oriented system designs as a background risk indicator rather than a standalone identity gate. The primary contribution lies in establishing a deployment-oriented modeling framework that aligns open-set behavioral plausibility assessment with resilient, human-centered security design principles for automation-intensive computing environments.

Acknowledgements

The authors thank Bálint Gál for his valuable assistance in programming and software development during the course of this research.

References

[1] D. Yu et al. A survey of anomaly intrusion detection techniques. *Journal of Computing Science and Engineering*, 2:428–448, 2004.

[2] F. Monrose and A. D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16:351–359, 2000.

[3] K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *Proc. IEEE/IFIP DSN*, pages 125–134, 2009.

[4] P. S. Teh et al. A survey on keystroke dynamics biometrics: Approaches, advances, and evaluations. *IEEE Access*, 10:1–28, 2022.

[5] M. L. Ali et al. Continuous authentication using keystroke dynamics. *Computers & Security*, 2023.

[6] S. Mondal and P. Bours. Liveness detection for behavioral biometrics: A survey. *IET Biometrics*, 2022.

[7] W. J. Scheirer et al. Toward open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1757–1772, 2013.

[8] A. Bendale and T. E. Boult. Towards open set deep networks. In *Proc. CVPR*, pages 1563–1572, 2016.

[9] H. Gonzalez et al. Synthetic keystroke dynamics for liveness detection. In *Proc. IEEE IJCB*, 2022.

[10] E. Bursztein et al. Easy does it: More usable CAPTCHAs. In *Proc. ACM CHI*, 2014.

[11] Y. Zhang et al. Automated input detection via temporal behavioral signals. *Future Generation Computer Systems*, 2024.

[12] A. Ometov et al. Multi-factor authentication: A survey. *Cryptography*, 2(1), 2018.

[13] J. Bonneau et al. The quest to replace passwords: A framework for comparative evaluation. In *Proc. IEEE S&P*, pages 553–567, 2012.

[14] V. Dhakal et al. Observations on typing from 136 million keystrokes. In *Proc. ACM CHI*, pages 1–12, 2018.

[15] T. Young et al. Keyboard layout effects on typing dynamics. *Applied Ergonomics*, 2018.

Enhancing LLM-Based Vulnerability Analysis Using Runtime-Aware Context: An Empirical Study on the Vul4J Dataset

Sami Abdel-Fattah , Judit Jász

Abstract: Large Language Models (LLMs) have demonstrated strong capabilities in software vulnerability analysis, yet most existing approaches rely exclusively on static source code. This limits their ability to capture the dynamic, runtime behavior through which vulnerabilities actually manifest. We present a runtime-aware enhancement pipeline that augments static code information with execution-derived evidence including failing test cases, error messages, and execution traces obtained via Proof-of-Vulnerability (PoV) execution on the Vul4J benchmark. We evaluate both a static baseline and our enhanced approach across four tasks: vulnerability detection, localization, severity assessment, and CWE description. Results show consistent improvements in severity estimation, and CWE name prediction quality, with ROUGE-L rising from 0.37 to 0.41 and severity accuracy improving from 0.89 to 0.94. These findings establish runtime context as a meaningful signal for LLM-assisted security analysis.

Keywords: software vulnerability analysis, large language models, runtime execution, Vul4J, CWE classification, program security

1 Introduction

In modern software engineering, software vulnerabilities continue to be one of the most persistent and significant risks. The need for automated tools that can detect, explain, and evaluate security flaws at scale has increased due to the adversarial nature of security research and the increasing complexity of code bases [1].

Promising methods for code-centric reasoning have been made possible by recent developments in large language models [2]. Large source code corpora can be used to train LLMs to detect suspicious patterns, determine intent, and produce natural language explanation capabilities that are directly applicable to vulnerability analysis tasks [3, 4]. However, in most existing approaches, the model is provided only with static source code, with no indication of the code's runtime behavior.

These signals provide insight into where and how the vulnerability manifests. Buffer overflows, use after free errors, and injection flaws are examples of vulnerabilities that frequently manifest during execution, for example in stack traces, failing assertions, or unexpected program states. A model that relies solely on syntax and structure might overlook the dynamic evidence that identifies the source and severity of a flaw.

This work aims to address this limitation. We propose and evaluate a runtime-aware augmentation approach, prior to querying the LLM, we execute the vulnerable program in the Vul4J Docker environment [5] using its Proof-of-Vulnerability test suite. The resulting runtime artifacts are then injected into the model's prompt along with the source code. Our primary objective is to evaluate whether this additional context leads to measurable improvements in analysis quality.

The contributions of this paper are:

- A runtime-aware augmentation pipeline for LLM-based vulnerability analysis that is reproducible and compatible with any instruction following LLM.
- A systematic comparative evaluation across four analysis tasks: detection, localization, severity assessment, and CWE description.
- Empirical evidence that execution context improves model outputs on the Vul4J benchmark, with discussion of practical implications and current limitations.

2 Methodology

2.1 Dataset

Eighteen vulnerable Java instances from the Vul4J dataset were selected for the experiments [5] based on reproducibility and execution stability within the Docker-based Proof-of-Vulnerability (PoV) environment. Several Vul4J projects required additional dependency configuration or produced incomplete runtime traces, making them unsuitable for consistent runtime-aware evaluation. To reduce potential selection bias, the selected subset spans multiple CWE categories and severity levels. Each instance includes a vulnerable source file, a PoV test case, a Docker environment, a human-written patch (ground truth for localization), a CWE identifier (ground truth for vulnerability description), and a compliance-based severity score (ground truth for severity assessment). The selected instances correspond to Vul4j IDs-1, 2, 6, 7, 13, 18, 20, 30, 35, 36, 39, 50, 60, 66, 69, 71, 72, 77.

2.2 Baseline Approach

The baseline configuration in this study is designed as a direct replication of the methodology proposed in [6], which introduces a multitask-based approach for vulnerability analysis using large language models. In the original work, the authors evaluate LLM performance on C++ source code. In contrast, our study adapts this baseline to Java code using the Vul4J dataset. This adaptation is an important contribution, as it allows us to examine the generalizability of the original approach across programming languages.

As illustrated in Figure 1, the LLM in the baseline setting receives only the vulnerable source code snippet along with a brief natural-language task prompt, without any additional contextual or execution-based information. The model is required to perform four tasks, as shown in Table 1: 1) detect whether the code contains a vulnerability, 2) localize the vulnerable function or region, 3) estimate a severity score on a scale from 1 to 10, and 4) predict the CWE identifier and corresponding vulnerability type.

No execution feedback or runtime information is provided in this baseline configuration. Therefore, this setup serves as a faithful reproduction of the original multitask framework, enabling a direct comparison between the baseline and the enhanced approach proposed in this work.

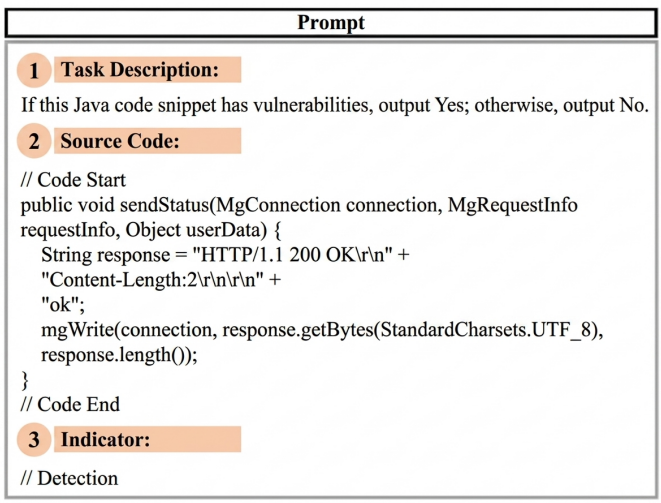


Figure 1: Prompt Example

Dimension	Task Description	Indicator
Vulnerability Detection	If this Java code snippet has vulnerabilities, output Yes; otherwise, output No.	// Detection
Vulnerability Assessment	Provide a qualitative severity ratings of CVSS v2.0 for the vulnerable Java code snippet.	// Assessment
Vulnerability Location	Provide a vulnerability location result for the vulnerable Java code snippet.	// Location
Vulnerability Description	Provide a CVE description for the vulnerable Java code snippet.	// Description

Table 1: Prompt components

2.3 Enhanced Approach

The enhanced approach extends the baseline by prepending runtime-derived evidence to the prompt. This evidence is collected by executing the vulnerable program against its PoV test suite inside the Vul4J Docker environment. The following artifacts are extracted and included, failing test case names and assertion messages, runtime error messages and exception stack traces, and high-level execution traces capturing the failing call path. These signals expose where and how the vulnerability manifests during execution, providing contextual grounding that purely syntactic analysis cannot supply.

2.4 Evaluation Procedure

Detection accuracy is reported as the proportion of instances where the model correctly identifies whether a vulnerability is present. Localization accuracy is evaluated as a function-level classification task. The model's prediction is considered correct if the identified function or method name exactly matches the ground-truth vulnerable function derived from human-written patches on GitHub. No distance-based metric (e.g., line-level or row-based distance) is used, as the evaluation focuses on function-level granularity. Severity assessment is evaluated using Mean Absolute Error (MAE) and accuracy within ± 1 point of the compliance-level reference. CWE description quality is measured using precision, recall, F1-score, and exact-match accuracy for CWE identifier classification, and ROUGE-1, ROUGE-2, ROUGE-L [7], and exact-match accuracy for CWE name similarity.

3 Results and Discussion

Both approaches achieved perfect detection and localization on all analyzed instances, reflecting the fact that all selected instances contain known vulnerabilities in the Vul4J dataset. More informative comparisons arise in the remaining tasks. Table 1 summarizes results across severity assessment (a) and CWE description, including CWE name (b) and CWE ID (c). The runtime-enhanced approach improves across nearly every metric. Severity MAE decreases from 0.94 to 0.89, and ± 1 accuracy rises from 0.89 to 0.94, suggesting that execution feedback helps the model calibrate perceived impact. CWE name prediction quality improves consistently across ROUGE variants (ROUGE-L: 0.37 $\rightarrow$ 0.41), and exact-match accuracy rises from 0.33 to 0.39. CWE ID F1 improves modestly from 0.33 to 0.35, while recall remains unchanged, indicating that runtime context primarily sharpens precision rather than broadening coverage.

These results suggest a consistent, if moderate, benefit from runtime augmentation. The improvements are not dramatic, which is expected given the small sample size and the inherent variability in LLM-generated outputs. Nonetheless, the directional consistency across all metrics strengthens confidence in the approach. A key limitation is the sample size, the current evaluation covers only a subset of Vul4J, and broader coverage including non-vulnerable

Metric	Base.	Enh.
MAE	0.94	**0.89**
Acc. (±1)	0.89	**0.94**

a.

Metric	Base.	Enh.
Ex. Acc.	0.33	**0.39**
Prec.	0.38	**0.39**
Rec.	0.33	0.33
F1	0.33	**0.35**

b.

Metric	Base.	Enh.
ROUGE-1	0.37	**0.41**
ROUGE-2	0.33	**0.39**
ROUGE-L	0.37	**0.41**
EM Acc.	0.33	**0.39**

c.

Table 2: Detailed comparison of Baseline and Enhanced approach.

instances would enable more robust conclusions and allow detection metrics to be meaningfully interpreted. Additionally, the quality of runtime artifacts varies by vulnerability type; some flaws produce sparse traces that add limited signal. Future work will address these limitations by expanding the dataset, incorporating non-vulnerable instances, and experimenting with more targeted runtime extraction strategies.

4 Conclusion

We presented a runtime-aware framework for LLM-based software vulnerability analysis, demonstrating that execution-derived context, failing tests, error messages, and stack traces improves the quality of vulnerability descriptions and severity estimation compared to a static-only baseline on the Vul4J benchmark. The improvements are consistent across metrics and methodologically reproducible. Future work will scale the evaluation to larger, balanced datasets and investigate how different types of runtime evidence contribute individually to analysis quality.

References

[1] Sheng, Z., Chen, Z., Gu, S., Huang, H., Gu, G. and Huang, J., 2025. LLMs in software security: A survey of vulnerability detection techniques and insights. ACM Computing Surveys, 58(5), pp.1-35.

[2] Thapa, C., Jang, S.I., Ahmed, M.E., Camtepe, S., Pieprzyk, J. and Nepal, S., 2022, December. Transformer-based language models for software vulnerability detection. In Proceedings of the 38th annual computer security applications conference (pp. 481-496).

[3] Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B. and Karri, R., 2025. Asleep at the keyboard? assessing the security of github copilot's code contributions. Communications of the ACM, 68(2), pp.96-105.

[4] Khare, A., Dutta, S., Li, Z., Solko-Breslin, A., Alur, R. and Naik, M., 2025, March. Understanding the effectiveness of large language models in detecting security vulnerabilities. In 2025 IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 103-114). IEEE.

[5] Bui, Q.C., Scandariato, R. and Ferreyra, N.E.D., 2022, May. Vul4j: A dataset of reproducible Java vulnerabilities geared towards the study of program repair techniques. In Proceedings of the 19th International Conference on Mining Software Repositories (pp. 464-468).

[6] Yin, X., Ni, C. and Wang, S., 2024. Multitask-based evaluation of open-source LLM on software vulnerability. IEEE Transactions on Software Engineering, 50(11), pp.3071-3087.

[7] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of the ACL Workshop on Text Summarization Branches Out.

A Multi-Criteria Scoring Model for Evaluating and Selecting Multi-Factor Authentication Mechanisms

Yusufbek Sulaymonov, Dr.Veronika Szücs

Abstract: Multi-Factor Authentication (MFA) is a cornerstone of modern cybersecurity, yet selecting appropriate authentication mechanisms remains a complex challenge due to trade-offs between security, usability, cost, and operational constraints. Existing research often evaluates MFA methods in isolation, lacking a unified quantitative framework for systematic comparison. This paper proposes a literature-informed multi-criteria scoring model grounded in a Multi-Criteria Decision-Making (MCDM) approach. Eight evaluation criteria—security, usability, phishing resistance, implementation cost, operational cost, compatibility, administrative overhead, and recovery resilience—are defined and weighted based on a structured synthesis of 26 peer-reviewed studies. The model is extended with criterion-specific contextual risk multipliers to support adaptive authentication in Zero Trust environments, and combined configurations are evaluated with a redundancy penalty computed from an explicit factor-class overlap matrix capturing diminishing security returns when methods share the same authentication class. The framework is explicitly designed as a transparent heuristic decision-support tool rather than an empirically calibrated predictive system. An author-constructed AHP contrast scheme identifies the same top-3 methods (TOTP, Push, Passkey) as the proposed model (top-3 set overlap = 100%); Spearman's $\rho = 0.57$ and Kendall's $\tau = 0.43$ are reported as point estimates of moderate agreement. Sensitivity analysis confirms top-three ranking stability in 97.6% of 5,000 Monte Carlo weight perturbations.

Keywords: multi-factor authentication; MCDM; risk-based authentication; Zero Trust; adaptive authentication; phishing resistance; decision support

1 Introduction

Multi-Factor Authentication (MFA) has become an essential layer of defense in digital security frameworks, substantially reducing the risk of unauthorized access [1]. Recent industry data underscores this importance: more than 99.9% of compromised accounts lacked MFA [2], and the average cost of a data breach reached $4.88 million in 2024 [3].

However, selecting and deploying MFA mechanisms introduces complex trade-offs. Biometric systems raise concerns related to privacy, revocability, and accessibility [4]. Possession-based systems are vulnerable to device theft and SIM-swap attacks [5]. Push-notification methods are increasingly exploited via MFA fatigue attacks, appearing in approximately 14% of analyzed security incidents [6, 7]. Only 28% of users have enabled MFA for even one account, with perceived complexity cited as a primary barrier [8].

This paper addresses three research questions. **RQ1:** What are the relative strengths and weaknesses of common MFA methods across multiple criteria simultaneously? **RQ2:** How can contextual risk be incorporated into MFA evaluation without requiring real-time empirical data? **RQ3:** How do combined MFA configurations compare with single-method deployments when factor overlap is accounted for?

The specific contributions are: (i) a structured synthesis of findings from 26 peer-reviewed MFA studies organized into eight explicit evaluation criteria with documented weight rationale; (ii) a redundancy-aware aggregation function that penalizes combinations sharing the same authentication factor class, computed from an explicit overlap matrix; (iii) criterion-specific risk multipliers enabling context-sensitive evaluation under Zero Trust principles; and (iv) a complete scoring rubric, literature-source registry, AHP contrast comparison, and Monte Carlo sensitivity analysis.

2 Background and Related Work

MFA requires credentials from at least two independent categories: knowledge (passwords, PINs), possession (tokens, smartphones), and inherence (biometrics), significantly improving resistance to unauthorized access [18, 19]. Despite its advantages, MFA does not guarantee security by itself; vulnerabilities arise from protocol flaws, implementation weaknesses, and human factors such as user fatigue and social engineering [21]. Recovery mechanisms represent an often-overlooked attack surface [13].

Comparative analyses consistently show that MFA methods differ in security strength, usability, deployment cost, and phishing resistance [22, 23]. Advanced mechanisms such as FIDO2 hardware tokens and passkeys offer strong cryptographic protection; CISA designates FIDO/WebAuthn as the gold standard for phishing-resistant MFA [14]. SMS and email-based OTPs remain widely deployed but are vulnerable to interception and social engineering [24, 25].

Existing evaluation approaches exhibit four recurring limitations: static weighting schemes ignoring contextual risk variation; no interaction modeling between factor-class overlapping methods; underrepresentation of recovery mechanisms and enrollment friction [13, 28]; and low interpretability in AI-based models [27]. MCDM techniques, particularly AHP, provide structure but assume static weights and do not model factor interaction [17, 26]. The present work addresses these gaps through context-aware, redundancy-aware multi-criteria evaluation.

3 Methodology

3.1 Design Philosophy

The proposed model is a *literature-informed heuristic MCDM framework*. All scores are derived from systematic synthesis of existing peer-reviewed studies and should be interpreted as relative comparative rankings rather than absolute security measurements. This design choice prioritizes interpretability, transparency of assumptions, and adaptability across organizational contexts.

3.2 Evaluation Criteria

Eight criteria capture both technical and user-experience dimensions. Weights L_i sum to $\sum_{i=1}^{8} L_i = 55$; the theoretical maximum score is $55 \times 10 = 550$.

1. **Security** ($L_1 = 8$): Resistance to unauthorized access and cryptographic robustness. Weighted highest alongside phishing resistance because security breaches represent the primary threat vector [9].

2. **Usability** ($L_2 = 7$): Ease of enrollment, authentication flow, and user acceptance. High weight reflects strong empirical evidence that usability barriers directly reduce adoption and compliance [10, 11, 12].

3. **Phishing Resistance** ($L_3 = 8$): Ability to withstand phishing and social engineering. Weighted equally to security following evidence that phishing is the dominant credential theft vector [6, 14].

4. **Implementation Cost** ($L_4 = 6$): Initial deployment and infrastructure requirements [25].

5. **Operational Cost** ($L_5 = 7$): Ongoing maintenance, support, and user help-desk costs.

6. **Compatibility** ($L_6 = 7$): Cross-device and cross-platform applicability.

7. **Administrative Overhead** ($L_7 = 6$): Level of required system and user management.

8. **Recovery Resilience** ($L_8 = 6$): Ability to restore access after credential loss without introducing new attack surfaces [13].

3.3 Scoring Model

Raw scores $x_{ij} \in \{1, \ldots, 10\}$ are assigned through a rule-based rubric (Appendix A) defining qualitative bands for each criterion with concrete literature anchors. The aggregated weighted score for method j is:

$$S_j = \sum_{i=1}^{8} L_i \cdot x_{ij} \tag{1}$$

The model follows the structure of a weighted utility function and satisfies standard properties: monotonicity (higher raw scores yield higher S_j) and dominance preservation (if method A dominates B across all criteria then $S_A \geq S_B$). Unlike classical AHP or TOPSIS, it additionally incorporates time-dependent risk weighting and non-linear aggregation for combined configurations through redundancy penalties.

Scores are normalized using the empirical maximum ($S_{\max} = 400$, achieved by TOTP):

$$N_j = \frac{S_j}{S_{\max}} \tag{2}$$

3.4 Factor-Aware Redundancy Model

To capture security overlap between authentication factors, a pairwise overlap matrix $O_{jk} \in [0, 1]$ is defined based on factor-class membership. Methods within the same narrow sub-type (e.g., two OTP-delivery channels) receive highest overlap (0.60); methods sharing a broad factor class but different sub-type receive moderate overlap (0.30–0.50); methods across different classes receive low overlap (0.05–0.10). The combined score with redundancy adjustment is:

$$S_{\text{comb}} = \sum_{j \in M} S_j - \beta \sum_{\{j,k\} \subset M} O_{jk} \cdot \min(S_j, S_k) \tag{3}$$

The $\min(S_j, S_k)$ term reflects the conservative principle that overlapping factors contribute no more than the weaker component. The redundancy coefficient $\beta = 0.15$ is calibrated so that the same-class penalty for two high-overlap methods is approximately 4.5% of the raw combined score—meaningful but not dominating.

3.5 Criterion-Specific Risk Adaptation

A global contextual risk signal $R(t) \in [0, 1]$ is computed from observable authentication signals: device trust $D(t)$, location anomaly $L(t)$, and behavioral deviation $B(t)$. Each criterion is assigned a sensitivity coefficient γ_i and the adaptive scoring function is:

$$S_j(t) = \sum_{i=1}^{8} L_i \cdot x_{ij} \cdot R_i(t), \quad R_i(t) = 1 + \gamma_i(R(t) - 0.5) \tag{4}$$

Table 1 shows γ_i values and the resulting multipliers at three representative risk levels. Security ($\gamma_1 = +0.40$) and phishing resistance ($\gamma_3 = +0.60$) are amplified under elevated risk, calibrated against NIST SP 800-63B AAL tier progressions [15]. Usability ($\gamma_2 = -0.40$) is symmetrically dampened. Operational criteria are held invariant ($\gamma_i = 0$): cost and compatibility do not structurally change with threat level.

Table 1: Sensitivity coefficients γ_i and risk multipliers $R_i(t)$.

Criterion	γ_i	**Low** $R{=}0.6$	**Med.** $R{=}0.8$	**High** $R{=}1.0$
Security (L_1)	$+0.40$	1.04	1.12	1.20
Usability (L_2)	-0.40	0.96	0.88	0.80
Phishing Resistance (L_3)	$+0.60$	1.06	1.18	1.30
Impl. Cost (L_4)	0	1.00	1.00	1.00
Operational Cost (L_5)	0	1.00	1.00	1.00
Compatibility (L_6)	0	1.00	1.00	1.00
Admin Overhead (L_7)	0	1.00	1.00	1.00
Recovery Resilience (L_8)	$+0.20$	1.02	1.06	1.10

3.6 Comparison with an AHP Contrast Scheme

To provide an alternative-weighting reference point, an author-constructed AHP pairwise matrix was specified using Saaty's 1–9 scale, deliberately emphasizing security-related criteria over operational criteria (grounded in NIST SP 800-63B and CISA guidance). This is characterized as an *author-constructed contrast scheme* rather than independent expert elicitation; structured elicitation from multiple domain experts is identified as priority future work. The AHP matrix yields a consistency ratio $CR = 0.007 < 0.10$ [17]. The resulting AHP weights assign Security + Phishing = 0.497, substantially more than the proposed model's normalized L_i (0.291), providing a meaningful contrast between balanced general-purpose and security-emphasized weighting philosophies.

3.7 Sensitivity Analysis

The scoring pipeline is deterministic; all inputs (L_i, x_{ij}, γ_i, O_{jk}, β) are explicitly tabulated, enabling full reproducibility. Each L_i was varied by $\pm 20\%$ one at a time (17 configurations total). For Monte Carlo analysis, all eight weights were independently perturbed by a factor drawn uniformly from $[0.80, 1.20]$ across $N = 5{,}000$ trials.

4 Scoring of MFA Methods

Table 2 presents the complete scoring matrix with all raw scores x_{ij}, weighted totals S_j, and normalized scores N_j for all eight evaluated methods. Score justifications are provided per cell in Appendix A.

TOTP achieves the highest score ($S_j = 400$, $N_j = 1.00$), reflecting strong balance across security, usability, and phishing resistance. Push Notifications rank second ($S_j = 397$) driven by high usability and recovery resilience. Passkeys ($S_j = 393$) demonstrate strong security and phishing resistance, slightly reduced by remaining compatibility gaps in legacy enterprise environments. Hardware tokens score lowest overall ($S_j = 314$) due to high costs, low compatibility, and poor recovery, despite the strongest security scores.

The password baseline achieves $N_j = 0.91$—apparently high for a single-factor method. This is an expected consequence of balanced weighting: passwords score 9 on five operational criteria (implementation cost, operational cost, compatibility, admin overhead, recovery) that together carry 58% of total weight. Under security-emphasized weighting ($L_1 = L_3 = 10$, all other $L_i = 2$), Password falls to last place, confirming the anomaly is a weighting artifact.

Table 2: Literature-informed multi-criteria scoring of MFA methods. All raw scores $x_{ij} \in \{1, \ldots, 10\}$ are shown. *Password-only is a single-factor lower-bound reference baseline, not a recommended deployment option.

Criterion	L_i	**SMS**	**Email**	**TOTP**	**Push**	**Passkey**	**HW Tok**	**Biom.**	**Pwd***
Security	8	6	5	8	6	9	9	8	3
Usability	7	7	8	6	9	8	5	9	7
Phishing Resistance	8	4	3	7	6	9	9	7	2
Impl. Cost	6	9	9	8	7	5	4	4	9
Operational Cost	7	8	8	8	7	6	5	5	9
Compatibility	7	9	9	8	9	7	4	7	9
Admin Overhead	6	6	6	7	6	5	4	4	8
Recovery Resilience	6	7	8	6	8	7	4	6	8
Score S_j	–	380	377	**400**	397	393	314	351	365
Norm. N_j	–	0.95	0.94	**1.00**	0.99	0.98	0.79	0.88	0.91

5 Results

5.1 Factor-Class Overlap Matrix

Table 3 presents the complete O_{jk} matrix. The OTP-delivery possession group {SMS, Email, TOTP, Push} receives high mutual overlap (0.60) because members share the fundamental attack surface of code-channel interception. Passkey and HW Token (both crypto-bound possession) overlap at 0.50. Biometric (inherence) has minimal overlap with possession methods (0.10).

Table 3: Factor-class overlap matrix O_{jk} (symmetric). Higher values indicate greater security redundancy between methods.

	SMS	**Em.**	**TOTP**	**Push**	**PK**	**HW**	**Bio**	**Pwd**
SMS	–	0.60	0.60	0.60	0.30	0.35	0.10	0.10
Email	0.60	–	0.60	0.60	0.30	0.35	0.10	0.10
TOTP	0.60	0.60	–	0.60	0.30	0.35	0.10	0.10
Push	0.60	0.60	0.60	–	0.30	0.35	0.10	0.10
Passkey	0.30	0.30	0.30	0.30	–	0.50	0.30	0.10
HW Tok	0.35	0.35	0.35	0.35	0.50	–	0.10	0.10
Biom.	0.10	0.10	0.10	0.10	0.30	0.10	–	0.05
Password	0.10	0.10	0.10	0.10	0.10	0.10	0.05	–

5.2 Combined MFA Configurations

Table 4 shows combined scores for representative configurations computed with $\beta = 0.15$. Same-class OTP pairs (SMS + Email, TOTP + SMS) receive the highest penalties. Passkey + TOTP (penalty 17.7) benefits from orthogonal security properties. Push + Biometric has the lowest penalty (5.3) as it spans possession and inherence.

Table 4: Combined MFA configurations with redundancy penalty ($\beta = 0.15$).

Configuration	**Raw**	**Pen.**	S_{comb}	N_{comb}
TOTP + Push	797	35.7	761.3	0.95
Passkey + TOTP	793	17.7	775.3	0.97
HW Token + Passkey	707	23.6	683.5	0.85
SMS + Email	757	33.9	723.1	0.90
TOTP + SMS	780	34.2	745.8	0.93
Push + Biometric	748	5.3	742.7	0.93
Password + TOTP	765	5.5	759.5	0.95
Passkey+Push+TOTP	1190	71.1	1118.9	0.93

5.3 Risk-Adjusted Scores

Table 5 shows scores under three risk levels using Eq. (4). Under high risk ($R = 1.0$), methods with strong security and phishing resistance scores (Passkey: $399 \rightarrow 422$) benefit disproportionately from amplified L_1 and L_3 weights. Password rises by only 4 points ($366 \rightarrow 370$) because its weak security (3) and phishing resistance (2) receive little leverage—the intended adaptive behavior.

Table 5: Risk-adjusted MFA scores under three contextual risk levels.

Method	**Low** $R{=}0.6$	**Med.** $R{=}0.8$	**High** $R{=}1.0$
TOTP	405	415	**425**
Push Notifications	400	407	413
Passkey	399	410	422
SMS OTP	383	388	394
Email	379	382	386
Biometric	355	363	372
Password (baseline)	366	368	370
Hardware Token	320	333	345

5.4 AHP Comparison

Under the security-emphasized AHP weighting, the ranking becomes: (1) Passkey (428), (2) TOTP (404), (3) Push (382), (4) Biometric (377), (5) HW Token (369), (6) SMS (351), (7) Email (333), (8) Password (298). Three rank-agreement statistics were computed between the proposed model and the AHP ranking: Spearman's $\rho = 0.57$ ($p = 0.14$); Kendall's $\tau = 0.43$ ($p = 0.18$); top-3 set overlap = **3 of 3 (100%)**. Neither correlation is statistically significant at $n = 8$; they are reported as point estimates of moderate agreement. The top-3 set overlap (TOTP, Push, Passkey under both schemes) is the practically relevant finding: a security architect would arrive at the same shortlist under either weighting philosophy.

5.5 Sensitivity Analysis

The top-three ranking (TOTP, Push, Passkey) remained stable in all **17 of 17 one-at-a-time** weight perturbation configurations (100%). Monte Carlo analysis ($N = 5{,}000$ trials, all eight weights perturbed $\pm 20\%$ simultaneously) preserved the top-three ranking in **4,882 trials (97.64%)**. The remaining 2.36% corresponded to coincident large downward perturbations on L_1 and L_3

displacing Passkey from rank 3. Redundancy coefficient sensitivity across $\beta \in [0.10, 0.20]$ preserved the relative ranking of all combined configurations.

6 Discussion

Rankings are inherently dependent on the chosen weighting scheme—the framework's core value is enabling stakeholders to explore trade-offs systematically. The redundancy mechanism captures *diminishing returns from overlap* but does not encode *absolute security of the weakest link*: Password + TOTP receives a low penalty (5.5) because the two methods belong to different factor classes, yet is security-inferior to passkey-based combinations. The risk-adaptive layer resolves this by penalizing low security and phishing scores under elevated risk. A more complete formalization would add an explicit weakest-link penalty term; this is identified as future work.

No single MFA method is optimal across all criteria. Under balanced weighting, usability-driven methods (TOTP, Push) rank highest. Under security-focused weighting or high-risk conditions, phishing-resistant methods (Passkeys, Hardware Tokens) dominate. Organizations should: (1) prioritize phishing-resistant technologies in high-risk environments; (2) adopt multi-factor combinations spanning distinct factor classes; (3) balance usability and security to ensure sustained adoption [12]; and (4) adopt adaptive MFA policies scaling authentication strength to contextual risk [16, 15].

Limitations. All scoring values are literature-derived estimates. Weight assignments reflect a general-purpose organizational baseline and may require expert elicitation for sector-specific calibration. The redundancy model is heuristic; a formal probabilistic model would require empirical attack-success data.

7 Conclusion

This paper introduced a multi-criteria scoring framework for evaluating MFA mechanisms, integrating eight explicitly weighted criteria into a unified model with redundancy-aware aggregation and contextual risk adaptation. All inputs are fully tabulated, enabling complete reproducibility. TOTP and Push Notifications perform best under balanced weighting; Passkeys dominate under security-focused or high-risk conditions. The integration of criterion-specific risk multipliers supports adaptive MFA strategies consistent with Zero Trust principles [15].

Appendix A: Complete Scoring Rubric

Table 6 provides the full scoring rationale for each criterion-method combination in Table 2.

Table 6: Complete scoring rubric with literature justification.

Criterion	Method	Score	Justification
Security	SMS OTP	6	Second factor over password; vulnerable to SIM-swap and SS7 attacks [8].
Security	Email	5	Email accounts frequently compromised; less reliable channel [25].
Security	TOTP	8	Time-bound codes resist replay; app-based storage reduces interception [22].
Security	Push	6	Vulnerable to MFA fatigue; no cryptographic origin binding [7].

Continued on next page

Table 6 – continued

Criterion	Method	Score	Justification
Security	Passkey	9	Cryptographic origin binding; private key never leaves device [14].
Security	HW Token	9	Hardware-bound keys; FIDO2 AAL3 compliant [15].
Security	Biometric	8	High inherent uniqueness; score reflects liveness detection variability.
Security	Password	3	Single factor; highly vulnerable to credential stuffing and phishing [2].
Usability	SMS OTP	7	Familiar UX; moderate friction from code transcription [8].
Usability	Email	8	Low learning curve; slightly slower delivery.
Usability	TOTP	6	Requires app installation and code reading; higher cognitive load [11].
Usability	Push	9	Single-tap approval; preferred in industry surveys [28].
Usability	Passkey	8	Device-native biometric/PIN; seamless once enrolled [14].
Usability	HW Token	5	Physical token requirement; difficult for users with disabilities [8].
Usability	Biometric	9	Seamless once enrolled; fast recognition; accessible [11].
Usability	Password	7	Universally familiar; cognitive load from complex policies and resets reduces UX.
Phishing Res.	SMS OTP	4	OTP relayable in real-time; SIM-swap bypasses entirely [25].
Phishing Res.	Email	3	Highly susceptible to email compromise and relay attacks.
Phishing Res.	TOTP	7	Time-bound reduces replay window; real-time relay still possible [6].
Phishing Res.	Push	6	Approval-based; vulnerable to fatigue attacks [7].
Phishing Res.	Passkey	9	Cryptographically bound to relying party origin; relay structurally impossible [14].
Phishing Res.	HW Token	9	Same cryptographic binding; CISA gold-standard [15].
Phishing Res.	Biometric	7	Device-local verification limits interception; spoofing risk varies.
Phishing Res.	Password	2	No protection; passwords are the primary phishing target [2].
Impl. Cost	SMS OTP	9	Relies on existing carrier infrastructure [8].
Impl. Cost	Email	9	Leverages existing email infrastructure.
Impl. Cost	TOTP	8	Free authenticator apps; low server-side complexity.

Continued on next page

Table 6 – continued

Criterion	Method	Score	Justification
Impl. Cost	Push	7	Requires push infrastructure and app.
Impl. Cost	Passkey	5	Legacy system integration requires additional effort [14].
Impl. Cost	HW Token	4	Hardware procurement; per-device cost significant at scale [25].
Impl. Cost	Biometric	4	Hardware-dependent; enrollment infrastructure adds cost.
Impl. Cost	Password	9	Near-zero; all infrastructure already exists.
Op. Cost	SMS OTP	8	Carrier fees at scale; low management overhead.
Op. Cost	Email	8	Minimal ongoing cost; email already maintained.
Op. Cost	TOTP	8	No per-authentication cost; app updates user-managed.
Op. Cost	Push	7	Notification infrastructure maintenance.
Op. Cost	Passkey	6	Long-term lower due to reduced helpdesk volume [14].
Op. Cost	HW Token	5	Replacement and inventory management costs.
Op. Cost	Biometric	5	Sensor maintenance; false acceptance/rejection handling.
Op. Cost	Password	9	Lowest direct cost; password reset helpdesk costs underestimated.
Compatibility	SMS OTP	9	Works on any mobile-capable device.
Compatibility	Email	9	Universal; requires only email access.
Compatibility	TOTP	8	Widely supported; requires authenticator app.
Compatibility	Push	9	Major platforms supported; requires smartphone.
Compatibility	Passkey	7	Broad platform support as of 2025–2026; legacy SSO gaps remain [14].
Compatibility	HW Token	4	USB/NFC requirements; limited on mobile-only systems [8].
Compatibility	Biometric	7	Device biometrics widely available; server-side less compatible.
Compatibility	Password	9	Universal; supported on every platform.
Admin	SMS OTP	6	Phone number management; number change updates required.
Admin	Email	6	Email account and recovery address maintenance.
Admin	TOTP	7	Seed key management; lower per-user admin than hardware [22].
Admin	Push	6	Device registration and deprovisioning.
Admin	Passkey	5	Credential management across devices; ecosystem complexity.

Continued on next page

Table 6 – continued

Criterion	Method	Score	Justification
Admin	HW Token	4	Procurement, distribution, loss replacement, inventory tracking.
Admin	Biometric	4	Template management; re-enrollment on sensor change.
Admin	Password	8	Reset handling; relatively mature tooling.
Recovery	SMS OTP	7	Phone number recovery available; SIM-swap risk during recovery.
Recovery	Email	8	Email recovery widely supported.
Recovery	TOTP	6	Backup codes required; users frequently lose these [13].
Recovery	Push	8	New device enrollment straightforward.
Recovery	Passkey	7	Cross-device sync improving; recovery paths maturing [14].
Recovery	HW Token	4	Physical loss requires admin intervention [13].
Recovery	Biometric	6	Re-enrollment on device change; permanent changes create edge cases.
Recovery	Password	8	Reset flows well-established; reset emails are an attack vector.

Appendix B: Literature-Source Registry

The 26 primary sources used to construct the scoring rubric and weight assignments are catalogued below, with the criteria each source most directly informs.

Table 7: Literature-source registry with primary rubric contribution.

Source	Type	Primary Contribution	Criteria
[1]	Journal	Foundational MFA taxonomy and factor classification	L_1, L_2, L_6
[5]	Journal	Mobile authenticator threat models; SIM-swap vectors	L_1, L_3
[18]	Journal	MFA review; comparative trade-offs	L_1, L_2, L_4
[19]	Journal	Modern MFA deployment trends	L_1, L_4, L_5
[20]	Journal	Layered authentication security analysis	L_1, L_3
[4]	Journal	Biometric privacy, revocability, and accessibility	L_1, L_2, L_8
[21]	Journal	MFA protocol-level vulnerabilities	L_1, L_3
[13]	Conf.	Recovery mechanism security; fallback weakness	L_8, L_1
[22]	Journal	Comparative scoring of MFA methods	L_1, L_2, L_3, L_4
[23]	Journal	Security analysis of authentication mechanisms	L_1, L_3

Continued on next page

Table 7 – continued

Source	Type	Primary Contribution	Criteria
[8]	Journal	User adoption survey; usability barriers; operational data	L_2, L_4, L_5, L_7
[24]	Journal	SMS/email OTP vulnerability survey	L_3, L_1
[25]	Journal	Real-time phishing relay analysis; SMB deployment data	L_3, L_4
[14]	Report	FIDO2/passkey specification and deployment data	L_1, L_3, L_6
[7]	Journal	MFA fatigue attacks empirical study	L_3, L_1
[26]	Journal	Fuzzy AHP for MFA in cloud	L_1, L_4
[27]	Journal	Fuzzy / ML-based authentication evaluation	L_1, L_2
[16]	Conf.	Context-aware adaptive MFA; Zero Trust mappings	L_1, L_3
[28]	Conf.	Large-scale user study; adoption friction	L_2, L_7
[11]	Journal	Usability benchmarks; authentication time and error rates	L_2, L_5
[9]	Journal	Security requirements and attack landscape	L_1, L_3
[29]	Conf.	Banking sector MFA deployment analysis	L_1, L_3, L_5
[2]	Report	Industry threat data; 99.9% compromised-accounts figure	L_1, L_3
[6]	Report	Breach statistics; phishing prevalence	L_3
[3]	Report	Breach cost data	L_1
[12]	Report	IT-professional survey; usability–security trade-off	L_2, L_7

References

[1] A. Ometov et al. Multi-factor authentication: A survey. *Cryptography*, 2(1), 2018.

[2] Microsoft. Microsoft Digital Defense Report 2023. `microsoft.com/security`, 2023.

[3] IBM. Cost of a Data Breach Report 2024. `ibm.com/security`, 2024.

[4] W. Yang et al. A comprehensive study of security and privacy guidelines, threats, and countermeasures. *J. Cybersecurity and Privacy*, 1(1):1–17, 2021.

[5] C. Lee. SIM swap attack: Awareness and prevention. *IEEE Security & Privacy*, 18(3):68–71, 2020.

[6] Verizon. 2024 Data Breach Investigations Report. `verizon.com/dbir`, 2024.

[7] P. Shrestha and N. Saxena. An offensive and defensive exposition of wearable computing. *ACM CSUR*, 56(2):1–33, 2023.

[8] A. Karim et al. MFA usability and adoption study. *IEEE Access*, 2024.

[9] K. Bock et al. Security requirements and attack landscape for modern authentication. *USENIX Security*, 2023.

[10] S. Das et al. The effect of social influence on security sensitivity. *SOUPS*, 2019.

[11] S. Furnell et al. Usability benchmarks for authentication. *Computers & Security*, 2019.

[12] JumpCloud. IT Trends Report 2024. `jumpcloud.com`, 2024.

[13] G. Gavazzi et al. Security analysis of account recovery mechanisms. In *Proc. IEEE EuroS&P*, 2022.

[14] FIDO Alliance. Passkeys: Overview and specifications. `fidoalliance.org`, 2024.

[15] NIST. SP 800-207: Zero Trust Architecture. National Institute of Standards and Technology, 2020.

[16] S. Kandula et al. Context-aware adaptive MFA in Zero Trust environments. In *Proc. IEEE CLOUD*, 2024.

[17] T. L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[18] S. Singh et al. A review of MFA mechanisms and trade-offs. *J. Information Security*, 2019.

[19] R. Ghifari et al. Modern MFA deployment trends. *Future Gen. Comput. Syst.*, 2025.

[20] A. Mohammed et al. Layered authentication security analysis. *Comput. Secur.*, 2023.

[21] J. Wee et al. MFA protocol-level vulnerabilities. *IEEE Trans. Inf. Forensics*, 2024.

[22] Author et al. Comparative scoring of MFA methods. *J. Cybersecurity*, 2024.

[23] A. Alimzhanova et al. Security analysis of authentication mechanisms. *IEEE Access*, 2024.

[24] T. Suleski et al. SMS/email OTP vulnerability survey. *Health Inf. Sci. Syst.*, 2023.

[25] T. Tran-Truong et al. Real-time phishing relay analysis. *Comput. Secur.*, 2025.

[26] R. Hersyah et al. FuzzyFortify: Fuzzy AHP for MFA in cloud. *IEEE Access*, 2025.

[27] Y. Chen et al. Fusion-based authentication evaluation. *Expert Systems with Applications*, 2025.

[28] S. Das et al. Why people are not using two-factor authentication at scale. *SOUPS*, 2019.

[29] M. Guri et al. Banking sector MFA deployment analysis. In *Proc. IEEE S&P*, 2023.

Data Ordering for Split-Based M-tree Construction

Roland Kotroczo, Janos Mark Szalai-Gindl

Abstract: In this paper, we present a distance-based data ordering strategy that improves the performance of M-trees on static metric datasets. The method consists of two main steps: (1) optimal selection of router objects for each level of the tree, and (2) sorting data points in an order that ensures that the pre-selected router objects become the actual router objects of the tree during tree construction. Experimental results show that the proposed approach significantly improves query performance compared to existing approaches. The measurements were performed on metric data where the distance distribution is similar to the distance distribution of uniformly distributed data. Specifically, the approach was evaluated on the Million Song Dataset, utilizing the number of distance comparisons as the primary performance metric.

Keywords: database; metric space; m-tree; indexing

1 Introduction

Similarity search and metric space indexing are foundational techniques for managing high-dimensional and non-vectorial data. The M-tree [1] dynamically indexes objects using only a metric distance function. However, its structure and performance heavily depend on the data insertion order [4]. Suboptimal insertion leads to overlapping internal nodes and tree imbalance, degrading search performance by increasing distance calculations and I/O operations. This issue is particularly prominent with static data, although bulk loading methods [5] and various split policies attempt to mitigate it.

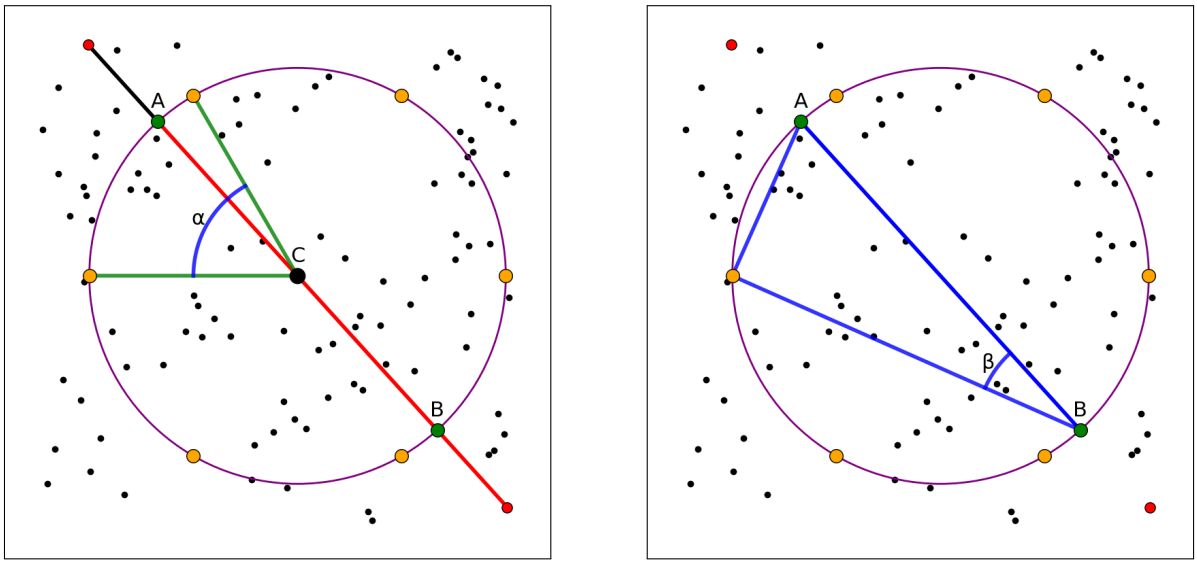
Similarity search and indexing in metric spaces rely on distance functions that satisfy the metric axioms, particularly the triangle inequality, to efficiently prune the search space during k-NN queries [3]. For general metric spaces where objects cannot be embedded in vector spaces, the M-tree [1] serves as a foundational dynamic, balanced index structure. It hierarchically partitions data into closed hyperspheres defined by routing objects and their coverage radii. Since M-tree search efficiency is highly sensitive to node splitting during insertion, various split strategies aim to minimize the overlap and volume of newly created routing regions. Rather than relying on these complex split heuristics, our approach utilizes the simple, heuristic-free FirstTwo split strategy [2], enhanced by our novel distance-based pre-ordering method applied prior to tree construction.

Our research investigates data ordering strategies to improve M-tree efficiency. While existing methods often rely on sampling-based clustering or space-filling curves [6], we focus on static data to develop a method that explicitly minimizes node overlaps.

Our main contribution is a distance-based pre-ordering method for static metric datasets. The approach consists of two steps: first, selecting the elements to serve as internal nodes for each tree level, and second, sorting the remaining elements to ensure their targeted placement during construction. Using a naive split method, our experiments demonstrate that this pre-ordering significantly improves M-tree search performance. The method is especially advantageous for uniformly distributed data or clustered data with similar internal distance distributions, offering a simple yet effective indexing solution for static datasets.

2 Method

A core principle of M-tree construction is minimizing overlap between routing objects. A naive approach to achieve this might be to always select the most distant pairs of points as routing centers during a split. However, while this minimizes overlap locally at a single level, it



(a) (b)

Figure 1: Selection of routing objects. The two mutually most distant points are marked in red. Points marked A and B are reference points taken on the straight line. The selected routing objects are marked in orange and the angles α and β required for their definition are indicated.

inadvertently forces the centers of adjacent subtrees at deeper levels to be placed too close to one another. This severely increases the global overlapping area across the tree, significantly degrading search performance.

2.1 Distance-based ordering

Our proposed method leverages the heuristic-free FirstTwo split strategy. The pre-construction ordering process consists of two phases: (1) selecting the routing objects to be used in the index structure, and (2) ordering these selected objects alongside the leaf-level data prior to insertion.

We construct the M-tree structure bottom-up. First, we identify the two most distant points in the dataset (highlighted in red in Figure 1a). Along the line connecting them, we define two reference points, A and B, located at $\frac{1}{6}$ and $\frac{5}{6}$ of the segment's length. We conceptually define a circle passing through A and B, upon which we will select our candidate routing objects (marked in orange).

Initially, we distribute $\lceil \frac{N}{M} \rceil$ points (M is the maximum node capacity and N is number of the points) evenly along this circle, separated by an angle $\alpha = \lceil 360/(\frac{N}{M}) \rceil$. We then assign each data point to its nearest candidate. If any routing object is assigned more than M points, we iteratively rotate all candidate points by 1 degree and re-evaluate. If a full rotation of $\alpha - 1$ degrees fails to satisfy the capacity constraint, we increment the number of routing objects and restart the assignment. If covering all points requires more than M routing objects, we create a new upper level in the tree and repeat the algorithm to cover the previous level's routing objects, continuing until the highest level contains at most M objects.

In vector spaces, the exact coordinates of these routing objects relative to the circle's center (C) can be easily determined (Figure 1a). However, in general metric spaces, the center cannot be precisely located. Instead, we rely strictly on distances to the fixed reference points A and B. By Thales' theorem, any point on the circle with diameter AB forms a right triangle with A and B. Thus, we can use the law of cosines to calculate the required angle β (where $\beta = \frac{\alpha}{2}$, as shown in Figure 1b) relative to A and B, allowing us to precisely identify the candidate routing objects using only the distance metric.

Once the hierarchical structure is defined, we determine the final insertion order using a level-order traversal of the constructed tree, prioritizing nodes destined for the upper levels. For each parent node, its assigned child nodes are inserted in ascending order of their distance from the parent. This level-by-level, distance-based sorting ensures that the relative order of points at level k preserves the structural order established at level $k-1$. Consequently, when the pre-ordered data is sequentially inserted, the FirstTwo split method naturally guarantees that our pre-selected points are perfectly utilized as routing objects from the bottom up.

2.2 Cluster-first approach

To improve data locality in larger datasets, we introduce a cluster-first approach. Clusters act as an additional, higher hierarchical level. Before applying the standard ordering process, we first extract and order the highest-level routing object (the root) from each cluster. If the total number of these top-level cluster representatives exceeds the page capacity M, we recursively group them. Once all cluster representatives are sequenced, the ordering continues with the standard level-by-level algorithm within each cluster. This ensures the final insertion order preserves both the parent-child relationships and the global cluster distribution.

3 Datasets

We evaluated our method using the Million Song Dataset (MSD) [7], which provides metadata and Echo Nest Analyze API tags (with relevance scores ranging from 0 to 100) for one million tracks. To enable tag-based clustering, we preprocessed the MSD by consolidating synonymous tags and categorizing them into 19 primary genres, assigning each a maximal relevance score of 100. Furthermore, we weighted the remaining tags based on their frequency of occurrence within the dataset and discarded any tags that could not be classified. Following the extraction and weighting of these features, we quantified the dissimilarity between any pair of tracks using the weighted Jaccard distance. Let T denote the universal set of tags, and let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{|T|}$ represent the tag relevance vectors for two given tracks. The distance d between them is defined as:

$$d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\sum_{t \in T} \min(x_t, y_t)}{\sum_{t \in T} \max(x_t, y_t)}$$

4 Results

To ensure an objective, hardware-independent evaluation, we measured search performance using the number of distance comparisons rather than query execution time in milliseconds, which is highly susceptible to system-level distortions. We evaluated our method on a cleaned subset of 7,403 MSD tracks. For the cluster-first approach, we applied DBSCAN [8] using a precalculated weighted Jaccard distance matrix. Parameter tuning yielded an optimal silhouette score of 0.85 (with $eps = 0.5$ and $min\_samples = 5$), resulting in 106 clusters and 436 outliers. The M-tree page capacity was set to 230 objects. We compared our pre-ordered FirstTwo method against the SamplingMinOverlapArea split strategy, which has been shown to achieve the best performance in comprehensive M-tree studies [2]. The results are summarized in Table 1.

5 Conclusions

Our measurements demonstrate that the proposed pre-ordering method outperforms existing alternatives. While applying the method globally to the entire dataset provides a clear performance gain, introducing a cluster-first approach yields even more substantial gains by enabling

Table 1: Average number of distance comparisons performed during searches on the MSD dataset

Picksplit methods	Clustered	Not clustered
FirstTwo	2333	4084
SamplingMinOverlapArea	-	4555

the local optimization of routing objects. Although we evaluated our approach using a Python prototype for rapid experimentation, the method can be architecturally integrated into complete database management systems. For instance, it can supplement existing M-tree implementations within PostgreSQL using the GiST framework [2, 9]. Future research will focus on generalizing the proposed data ordering strategy and exploring its applicability to datasets with varying spatial distributions.

References

[1] Ciaccia, P., Patella, M., & Zezula, P. (1997). M-tree: An Efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd VLDB conference, Athens, Greece* (pp. 426–435).

[2] Donkó, I., Szalai-Gindl, J. M., Gombos, G., & Kiss, A. (2019). An implementation of the m-tree index structure for postgresql using gist. In *2019 IEEE 15th International Scientific Conference on Informatics* (pp. 189–194). IEEE.

[3] Chen, L., Gao, Y., Song, X., Li, Z., Zhu, Y., Miao, X., & Jensen, C. S. (2022). Indexing metric spaces for exact similarity search. *ACM Computing Surveys*, 55(6), 1–39.

[4] Mao, R., Xu, W., Singh, N., & Miranker, D. P. (2005). An assessment of a metric space database index to support sequence homology. *International Journal on Artificial Intelligence Tools*, 14(05), 867–885.

[5] Loh, W.-K. (2021). Efficient Flexible M-Tree Bulk Loading Using FastMap and Space-Filling Curves. *Computers, Materials & Continua*, 66(2).

[6] Bercken, J., & Seeger, B. (2001). An evaluation of generic bulk loading techniques. In *VLDB*.

[7] Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). The million song dataset. In *11th International Conference on Music Information Retrieval (ISMIR)*.

[8] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD* (Vol. 96, No. 34, pp. 226–231).

[9] Hellerstein, J. M., Naughton, J. F., & Pfeffer, A. (1995). *Generalized search trees for database systems*. September.

A High-Performance GPU-accelerated Parallel 2D Multigrid Poisson Solver

Balint Toth, Zoltan Juhasz

Abstract: Particle-in-Cell simulation is an important tool in plasma science, where certain properties and behaviour can only be examined by simulations. Due to the large number of particles and simulation cycles, these simulations are extremely time-consuming and can only be executed in acceptable time with parallel implementations. A crucial step in the simulation is the solution of the Poisson equation to calculate the electric field that provide the basis of interactions among the particles. In this paper, we describe the design, implementation and performance optimisation of a GPU-based multigrid Poisson solver for two-dimensional simulations. Compared with a sequential C implementation, the GPU multigrid version achieved from 5.5 – 44.8x speedup for the frequently used grid sizes of 255×255 to 1023×1023.

Keywords: Poisson equation, Multigrid, GPU, CUDA, plasma simulation

1 Introduction

Solving the Poisson equation is a fundamental step in Particle-in-Cell (PIC) plasma simulations, where interaction among particles, and consequently their movement, is computed via the particle-generated electrostatic field [1] . The plasma is represented by a large number (10^5–10^8) of charged super-particles (electrons and ions) that move around due to high-frequency driving voltage. The (one-, two- or three-dimensional) space is discretised into a grid where the grid points will accumulate charges of the particles. The fundamental PIC cycle consists of four principal steps: (1) interpolation of particle charges onto the grid to obtain charge density, (2) solution of Poisson's equation $\nabla^2\phi = \rho/\varepsilon_0$ to determine the electrostatic potential, (3) computation of electric fields and interpolation back to particle positions, and (4) integration of the equations of motion to advance particle velocities and positions. When collisional effects are important, Monte Carlo collision algorithms are incorporated to model interactions between charged species and neutrals, as well as surface processes [1]. In two- and three-dimensional simulations, solving the Poisson equation is the most time-consuming step of the simulation. In this paper, we describe the implementation, optimization and performance results of a general-purpose, GPU-based parallel multigrid Poisson solver intended to be used in the simulation of two-dimensional low-temperature plasmas.

2 Multigrid Poisson equation solver

The simplest traditional method to solve the Poisson equation is the Jacobi iteration relying on finite difference equations. A new value at each grid point is computed by taking the average of the neighbouring four grid points. The process is iteratively repeated until the change of values at the grid points falls below a predefined threshold, ε. Two further variants, the Gauss-Seidel and the Successive Over Relaxation (SOR) methods are also widely used for solving PDEs. However, due to their unfavourable convergence rate, all these algorithms require thousands of iterations for the solution making them a performance bottleneck in plasma simulation programs. Fast direct solver methods exist in the form of spectral solvers that use the Fourier, Sine or Cosine transform, but their use is limited to problems with specific boundary conditions and simple, rectangular geometries.

Multigrid is a collective term for a form of iterative solvers that rely on repeatedly coarsening the grid representing the problem domain to eliminate low frequency error terms faster, which

in turn significantly improves convergence rates. Multigrid methods leverage the smoothing property of iterative solvers in conjunction with coarsening to speed up the elimination of the low-frequency components of the error.

The series of steps within multigrid methods is often referred to as a V-cycle (see Fig. 1). Other cycles also exist based on different coarsening strategies (e.g., F-cycle, W-cycle), but our implementation currently focuses on the V-cycle as described by Algorithm 1. In the descending phase of each V-cycle, the following operations are performed: (1) an initial approximation is computed and smoothened using a basic Jacobi iteration, (2.2) the residual equation $r = \rho/\varepsilon_0 - \nabla^2\phi$ is computed using a discrete approximation, (2.3) then the problem is restricted to a coarser grid half the size of the original. After the recursive application of steps (1) – (2.4) we reach a 3×3 grid size on which the Poisson equation can be solved directly (2.1). Subsequently, in the ascending phase of the cycle, the coarser grid is then prolongated back to the original problem size by (2.5) interpolating the missing grid points that were previously eliminated with the restriction and (3) applying another smoothing step. The rigorous mathematical description of these steps can be found in [2]. Our algorithm currently executes a pre-set number of V-cycles based on estimations of convergence for different grid sizes. The Full Multigrid Cycle (FMG) and the residual error-based iteration stopping condition will be implemented in the near future.

Algorithm 1 Pseudocode of the Multigrid solver

```
procedure V-CYCLE
    Error smoothing                                  ▷ (1)
    if on the coarser grid then                      ▷ (2)
        Compute exact solution                       ▷ (2.1)
    else
        Compute residual                             ▷ (2.2)
        Restrict to coarser grid                     ▷ (2.3)
        call V-CYCLE(Coarser grid)                   ▷ (2.4)
        Prolongate to finer grid                     ▷ (2.5)
    end if
    Error smoothing                                  ▷ (3)
end procedure
```

As the first step of the implementation process, a sequential multigrid solver was developed in C++ to serve as a baseline. This implementation was based on the recursive Algorithm 1. The code was designed to work with irregular grid resolutions and rectangular geometries. This sequential solver could have been used in the plasma simulation program but since every other step of the simulation runs on the GPU [3], this would have required extensive data transfer operations to copy solver input data from the GPU and the electric field data back to the GPU. Unfortunately, the latency and bandwidth limits of the PCI-e bus introduce large overheads and inefficiencies in the simulation making this hybrid CPU-GPU approach inefficient. To mitigate this, we implemented a highly-parallel GPU version of the multigrid algorithm that eliminates the need for extensive host-device data transfers and provides further execution time reduction due to the parallel GPU execution.

3 GPU-based Multigrid solver

The first version of the GPU-parallel multigrid implementation used the baseline CPU code as a skeleton where each individual operation within the V-cycle was replaced with a GPU kernel written in the Nvidia CUDA programming model. Each grid point was mapped to a GPU thread and operations on the grid points execute in parallel on the GPU. The implementation

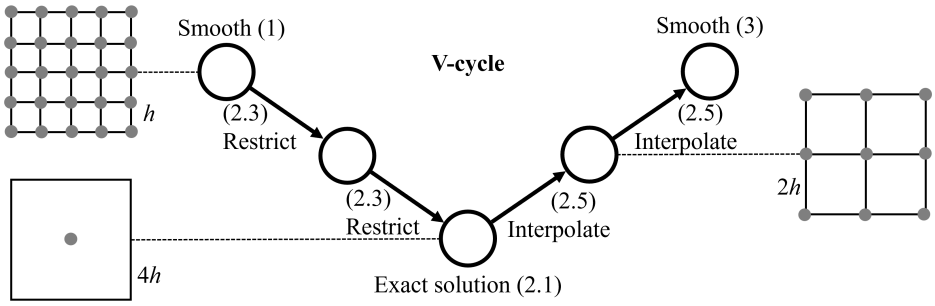


Figure 1: Illustration of the V-Cycle

Table 1: Evaluation of the different V-cycle GPU implementations for a 255×255 grid

V-Cycle implementation	Number of kernels	V-cycle time	Useful computation time
Naïve (RTX6000 Ada)	55	670 µs	127 µs
Fused kernel (RTX6000 Ada)	22	200 µs	76 µs
Fused kernel (A100)	22	114 µs	92 µs

uses only GPU memory, eliminating host-device data transfers during the simulation cycle. Since the Jacobi iteration overwrites the grid it is operating on, we used double buffering for the smoothing steps. Some runtime characteristics of a single GPU V-cycle are shown in Table 1 for 255×255 grids.

As the first row of Table 1 illustrates, the implementation launched many kernels, the launch overhead as well as GPU memory read and write operations result in low efficiency as the sum of the kernel runtimes is only 20% of the total V-cycle time. Note, that as the Multigrid algorithm gradually reduces the grid size and falls below a 32×32 grid, the problem can be handled in a single CUDA thread block eliminating the need for inter-block synchronisation. As a result, we designed and implemented an improved V-cycle that runs in a single fused kernel without recursion for the grid sizes below 32×32. This modification resulted in a near 10x speedup compared to the naïve, multiple-kernel version for smaller grids. With the use of a strided memory access pattern, this method is theoretically capable of running the entire multigrid algorithm, although with limited use of parallel computing. In order to leverage the speedup provided by the fused kernel, we implemented a combined algorithm that starts with individual kernels for each operation on large grids and switches to the fused kernel below 32×32. As can be seen in the second and third rows of Table 1, the improved version achieved significantly better GPU utilisation of up to 80%.

Since the improved V-cycle implementation is made up of 22 kernel calls and we need at least 20 V-cycles for convergence, the execution trace is dominated by kernel launch operations. We further optimised this step by using CUDA Graph technology that captures the kernels in one V-cycle as a task graph that can be subsequently launched with one call, significantly reducing the launch overheads.

4 Results

The numerical correctness of the developed multigrid algorithms were compared with reference implementations. This included a multigrid implementation written in Matlab [4] and the existing Jacobi iterative and Sine-transform based direct solvers from our plasma simulator codebases [1]. Both the C++ and the CUDA implementations achieved an accuracy in the order of 10^{-9} with 20 V-cycles compared to the baseline.

Table 2: Comparison of the execution times of one V-cycle in different implementations.

Grid size	Sequential baseline	CUDA Multigrid			
		RTX6000 Ada		A100	
	time [μs]	time [μs]	speedup	time [μs]	speedup
255×255	569.94	120.16	4.7	103.90	5.5
511×511	2550.70	184.35	13.8	146.13	17.5
1023×1023	12195.24	391.35	31.2	272.48	44.8

The runtime performance of the optimised GPU multigrid algorithm was measured on two different Nvidia GPU cards (RTX6000 Ada and A100) and compared with that of the sequential baseline implementation measured on an Intel i7-9700K 3.6 GHz CPU. Table 2 lists the achieved execution times for a single V-cycle of the different implementations. The optimised GPU algorithm running on the A100 card performed the best on grid sizes frequently used in plasma simulations (from 255×255 to 1023×1023) achieving speedups in the range from 5.5–44.8x.

5 Conclusions

In this paper, we presented the results of our effort to design and implement a high-performance, GPU-accelerated two-dimensional Multigrid Poisson solver which is a critical part of Particle-in-Cell plasma simulations. The implementation achieved up to 44.8x speedup over the sequential C++ version, significantly reducing the overall simulation time. We have applied various optimisation techniques to increase the efficiency of the implementation. The accuracy of the final version was validated against reference implementations. The performance measurements highlighted that more affordable GPU cards also provide high performance; efficient simulation is possible without supercomputer-class GPUs. The implementation is already integrated into several 2D plasma simulation codes and it opens the way for the development of efficient 3D PIC/MCC plasma simulations.

Acknowledgements

Balint Toth was supported by the Ministry of Culture and Innovation of Hungary through the University Research Fellowship Programme (EKÖP, Code: 2025–2.1.1–EKÖP–2025–00029/46) of the National Fund for Research, Development and Innovation. We acknowledge the support of DKF Ltd. for providing access to the Hungarian supercomputer Komondor. We also acknowledge the support of the NVIDIA Academic Grant Program.

References

[1] A. T. Powis et al., "Benchmark for two-dimensional large scale coherent structures in partially magnetized E × B plasmas - Community collaboration & lessons learned." *Plasma Sources Sci. Technol.*, vol. 35, no. 2, p. 025002, Jan. 2026, doi: 10.1088/1361-6595/ae3985.

[2] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, Second Edition.* SIAM, 2000.

[3] Z. Juhasz, J. Ďurian, A. Derzsi, Š. Matejčík, Z. Donkó, and P. Hartmann, "Efficient GPU implementation of the Particle-in-Cell/Monte-Carlo collisions method for 1D simulation of low-pressure capacitively coupled plasmas," *Comput. Phys. Commun.*, vol. 263, p. 107913, Jun. 2021, doi: 10.1016/j.cpc.2021.107913.

[4] D. Appelhans, "MultiGridMatlab." https://github.com/dappelha/MultiGridMatlab (accessed Mar. 01, 2026).

Hybrid bitset-based data types for optimizing runtimes of set operations

Bertalan Kovács, Botond Bertók, Márton Frits

Abstract: Efficient computation of set operations, such as **union**, **intersection**, and **difference**, is a fundamental operation in many domains. For dense data over a fixed-size universe, bitset representations are often in practice, as they are typically the most efficient approach when attempting to minimise algorithm runtimes. In this paper, we propose a method that maintains the efficiency of the bitset representation for dense data while improving performance for sparser collections. The approach dynamically switches the underlying container representation when the container is sparse, mitigating the primary drawbacks of bit array-based storage. We evaluate alternative container replacements from the C++ standard library and perform a comparative performance analysis to determine an effective threshold for switching representations. Experimental results show that the proposed method achieves performance comparable to bitsets in the dense case while significantly improving runtimes for sparse data, closely matching theoretical expectations across the full range of densities.

Keywords: Dense and sparse data, Data structures, Bitsets, Hybrid data structures, Runtime optimization

1 Introduction

Many applications operate on sets whose elements come from a fixed, known universe, including constraint solvers and graph algorithms. In such settings, the universe size is known in advance and remains constant during computation. Bitsets are often used for fixed-universe sets, as they allow for efficient implementations of fundamental set operations such as union, intersection, and difference.

When a large fraction of the universe is present in the set, bitsets are one of the most efficient general-purpose representations in practice[4]. However, the runtimes of bitset operations solely depend on the size of the universe and are independent of the actual number of stored elements.

When sets are sparse, this behaviour is inefficient. In contrast, representations whose runtimes scale with the number of stored elements can offer significantly better performance when working with sparse containers. This observation suggests that a hybrid container applying both approaches could negate the main drawback of bitset representations by using an array-based container for sparse data.

In this paper, we propose a hybrid representation for fixed-universe data that combines the efficient operation of bitsets for dense sets with a more efficient sparse representation when the number of elements is small. By dynamically selecting the underlying container based on set density, our approach avoids the inefficiencies of bitsets in the sparse case while preserving their advantages for dense data. We evaluate this strategy empirically and show that it follows the minimum of the two underlying storage types across a wide range of set densities.

2 Related work

Bitsets enable very fast set operations through word-level parallelism, but their space cost is proportional to the universe size, making them inefficient for sparse data. Database and information retrieval systems have long addressed this limitation through compressed bitmap schemes such as WAH[1] and EWAH[2], which use run-length encoding to reduce storage while still allowing logical operations directly on compressed data. These approaches aim not only to improve space usage but also to preserve fast intersection and union operations without full decompression, thereby keeping runtimes low even for large universes.

Another line of work mitigates sparsity by adapting the representation itself. More recent designs, such as Roaring bitmaps[3], partition the universe into fixed-size chunks and select a representation per chunk (e.g., array containers for sparse regions and bitmap containers for dense ones). This localised adaptability preserves word-level efficiency where beneficial while avoiding the runtime and memory overhead of global sparse bitmaps. Together, these approaches seek to retain the computational advantages of bitsets while minimising the performance penalties that arise for sparse data.

3 Sparse Representation Selection

To complement the bitset-based representations in sparse containers, we require a container whose performance scales with the number of elements stored rather than with the size of the universe. One of the most important factors when choosing this replacement is that the replacement implementation must support efficient iteration and minimal memory allocation overhead, as these dominate the runtime in sparse cases. We evaluate three candidates from the C++ standard library: ordered sets (`std::set`), hash-based sets (`std::unordered_set`), and dynamic arrays(`std::vector`).

To minimise the runtime of set operations – particularly merge-based algorithms – the primary selection criterion is element access performance. In the evaluated containers, dynamic arrays provide constant-time random access ($\mathcal{O}(1)$), while ordered sets and hash-based sets require logarithmic or amortised constant-time access with significantly higher constant factors. Consequently, we select `std::vector` as the replacement container.

4 Determining threshold

In theory, the crossover point can be estimated analytically. Since vector-based operations require approximately $2N$ element comparisons ($T_v = 2N$), while bitset-based operations require processing all K bits ($T_b = \frac{K}{64}$ for 64-bit architectures), using $N = K\delta$ the two runtimes become equal when

$$T_v = T_b \Rightarrow \delta = \frac{1}{128}$$

This corresponds to a theoretical density threshold of $\delta_{\text{theoretical}} \approx 0.0078$.

To validate the theoretical threshold in practice, we measured the runtimes of set operations implemented using merge-based algorithms on dynamic arrays[1], and compared them against bitset-based implementations on uniformly distributed data with a universe size of 5×10^6. The empirical density threshold was determined as the point at which the runtimes of the two approaches were equal.

Based on these measurements, the practical density threshold was significantly lower ($\delta_{\text{empirical}} \approx 0.003$). The difference between the analytical and empirical thresholds can be attributed to memory hierarchy effects and constant-factor differences ignored by the theoretical model. While vector-based operations access only the stored elements, bitset-based operations must scan the entire universe, resulting in memory-bandwidth-bound behaviour once the working set exceeds the effective low level, faster cache capacity. Additional benchmarks were run that measured the crossover thresholds across varying universe sizes and densities to confirm this behaviour. The results show that bitset runtimes remain nearly constant with respect to density but scale with universe size, whereas vector runtimes scale linearly with the number of stored elements, irrespective of container densities. As the universe size increases, the empirical crossover density decreases, indicating that memory hierarchy effects dominate the practical threshold.

[1] Using `std::set_union`, `std::set_intersection`, and `std::set_difference`

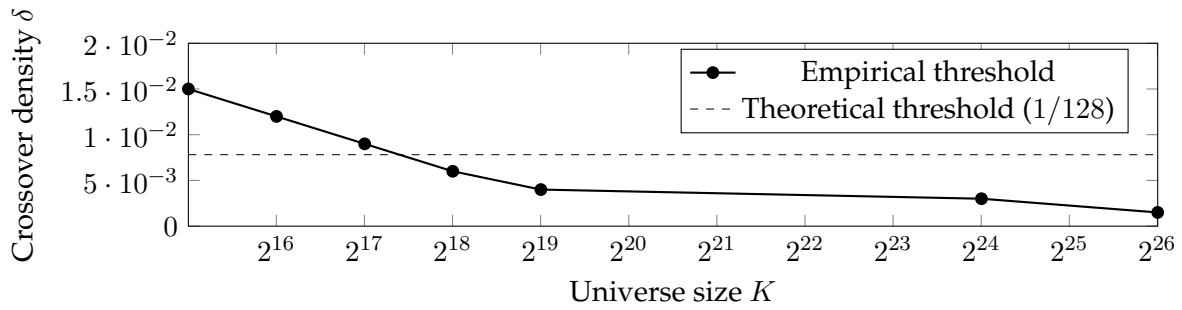


Figure 1: Empirical crossover density as a function of universe size K. The theoretical threshold ($\delta = 1/128$) is shown as a dashed horizontal line. As K increases, the practical crossover density decreases due to memory hierarchy effects.

5 Technical details

To avoid the overhead of frequent conversions during dynamic resizing, the container representation is fixed at initialization. For a universe size K with a chosen threshold δ, the maximum memory overhead is $K \times |1 - 64\delta|$ bits, where the constant 64 accounts for the 64-bit system architecture. Practically, this consumption remains strictly bounded: it is capped by the requirements of the bitset representation in larger universes and never exceeds those of the vector representation in smaller universes.

The input elements are predefined and sorted before measurements begin, as the operating units and materials in the p-graph problem are known prior to executing the solver. Consequently, we leveraged `std::set_difference`, `std::set_union`, and `std::set_intersection` from the STL for homogeneous container types. For operations between disparate types (vector vs. bitset), we used manual iteration over the vector elements to maintain performance.

6 Runtime results

For a universe of size 5×10^6, the empirical density threshold was approximately $\delta \approx 0.003$, corresponding to the point at which the vector-based and bitset-based runtimes were equal. The proposed hybrid approach closely tracks the lower envelope of the two performance curves: for sparse inputs, its runtime matches that of the vector-based implementation, and as the density approaches the threshold, it transitions smoothly toward the bitset runtime.

This confirms that automatic switching between representations effectively avoids the performance degradation that can be experienced when using only one of the presented methods. Small fluctuations in runtime are likely due to cache effects and system noise rather than algorithmic instability.

In the reported experiments, no representation conversion was performed after the set operations were completed. This reflects practical usage, as converting the result between vector and bitset representations would require not only an additional full traversal of the data but also the allocation of a new container instance. In particular, converting from a sparse vector to a bitset includes allocating memory proportional to the size of the universe, which would exceed the memory footprint of the original sparse representation. Such conversion costs can therefore dominate the total runtime and negate the performance benefits obtained by selecting the optimal representation for the set operation itself.

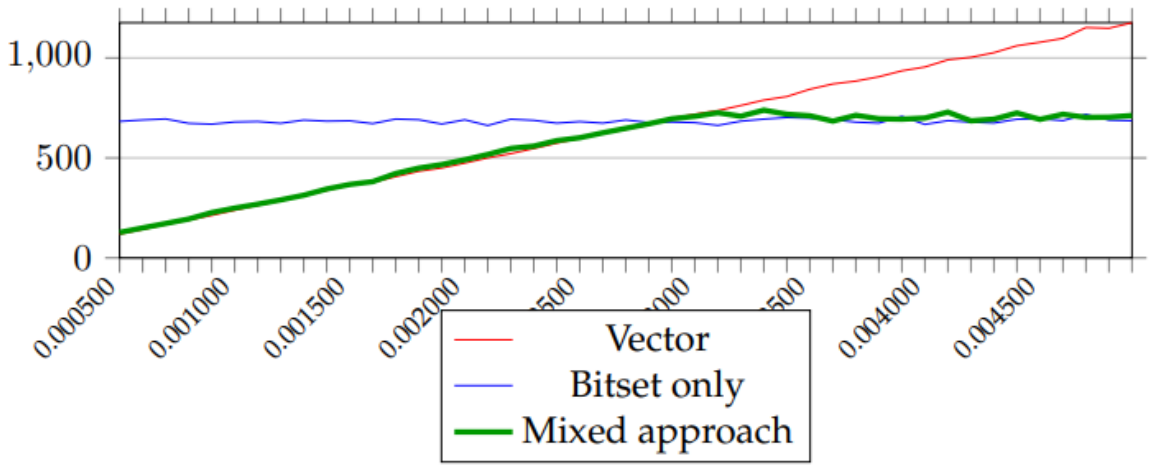


Figure 2: Runtimes with uniform inputs

7 Conclusion

This work examined the performance trade-offs between vector-based and bitset-based representations for set operations under varying data densities. While analytical modelling suggests a theoretical crossover density of $\delta = \frac{1}{128}$, empirical measurements demonstrate that the practical threshold is significantly influenced by constant factors and memory hierarchy effects ignored by the simple theoretical model. By automatically switching between representations at an empirically determined density threshold, the proposed hybrid approach consistently achieves performance close to the minimum of the two individual implementations. In sparse regimes, it matches the efficiency of merge-based vector operations, while in dense regimes, it benefits from the word-level parallelism of bitset operations. This confirms that adaptive representation selection effectively avoids the performance degradation associated with using a single representation across all densities.

Although the hybrid strategy significantly improves practical performance, further gains may be achievable by employing alternative sparse representations, such as compressed or Roaring-style bitmaps. These data structures combine aspects of both dense and sparse storage and may reduce memory traffic while retaining efficient set operations. Exploring such representations within the adaptive framework presents a promising direction for future work.

References

[1] Wu, K., Otoo, E. J., & Shoshani, A. (2006). Optimizing bitmap indices with efficient compression. *ACM Trans. Database Syst.*, 31(1), 1–38.

[2] Lemire, D., Kaser, O., & Aouiche, K. (2010). Sorting improves word-aligned bitmap indexes. *Data & Knowledge Engineering*, 69(1), 3–28.

[3] Chambi, S., Lemire, D., Kaser, O., & Godin, R. (2015). Better bitmap performance with Roaring bitmaps. *Software: Practice and Experience*, 46(5), 709–719.

[4] Colantonio, A., & Di Pietro, R. (2010). Concise: Compressed 'n' Composable Integer Set. *Information Processing Letters*, 110(16), 644–650.

Bounded Exact Channel-Native Fusion of Local Noisy Motifs in Density-Matrix Simulation

Zoltán Kégli, Tamás Kozsik, Péter Rakyta

Abstract: Exact noisy quantum-circuit simulation remains expensive, yet many structured workloads contain small local noisy regions whose evolution may be reusable if their open-system semantics are preserved exactly. We study a bounded form of that problem inside the `SQUANDER` density-matrix stack: same-support 1- and 2-qubit gate-noise motifs are promoted to composable completely-positive trace-preserving (*CPTP*) objects represented primarily as Kraus bundles, and validated against sequential density-matrix evolution. The method separates a **strict** motif-proof layer from an explicit **hybrid** whole-workload layer. Implementation-backed evidence reaches both layers: six counted cases (four microcases and two continuity anchors at 4 and 6 qubits) match the sequential reference within Frobenius norm $\leq 10^{-10}$; a five-case external slice agrees with an independent density-matrix simulator to $\approx 10^{-15}$; and a frozen 26-case whole-workload matrix classifies 17 workloads as already served by the existing fusion baseline, 9 as routing channel-native work that does not yet clear the speedup threshold, and 0 as channel-native-justified. Exact noisy fusion is therefore feasible and auditable on the bounded support surface studied here, while workload-level acceleration remains unresolved and must be argued separately from correctness.

Keywords: quantum circuit simulation, density matrix, CPTP channels, Kraus representation, noisy circuit fusion, circuit partitioning, SQUANDER

1 Introduction

Partitioning and gate fusion are standard tools in unitary quantum simulation [7, 8, 9, 10], while open-system theory provides the language of Kraus maps, Choi matrices, and Liouville superoperators for noisy evolution [2, 3]. What remains underdeveloped is the bridge between the two: how to treat a small noisy region of a circuit as an *exact reusable object* without losing the ordered semantics of open-system evolution. Exact density-matrix simulation [4, 5, 6] provides the cleanest reference for noisy variational algorithms, but its cost forces careful choices about what structure can be reused safely. The present study gives a bounded answer: 1- and 2-qubit same-support noisy motifs can be composed as exact CPTP objects and validated against sequential density-matrix evolution, rather than being treated only as barriers between unitary islands. The contribution is methodological. A scientifically credible path needs both a strict motif-proof layer, where the fused object itself is validated, and a separate hybrid whole-workload layer, where that object is tested inside larger exact workloads with explicit route attribution. The current evidence reaches both layers; the matrix-level performance picture is mixed.

2 Problem Formulation and Notation

Let a noisy circuit on n qubits be an ordered list of operations $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_L$, where each $\mathcal{O}_i$ is either a unitary gate U_i or a local CPTP channel $\mathcal{N}_i$ acting on a qubit subset $S_i \subseteq \{0, \ldots, n-1\}$ (each operator acts as the identity on the untouched qubits). The reference (sequential) evolution of a density matrix $\rho \in \mathbb{C}^{2^n \times 2^n}$ is

$$\rho_{\text{seq}} \;=\; (\mathcal{E}_L \circ \mathcal{E}_{L-1} \circ \cdots \circ \mathcal{E}_1)(\rho_0), \quad \mathcal{E}_i(\sigma) = \begin{cases} U_i\, \sigma\, U_i^\dagger & \text{(gate)} \\ \sum_k K_k^{(i)}\, \sigma\, K_k^{(i)\dagger} & \text{(channel)} \end{cases} \tag{1}$$

A *motif* is a contiguous subsequence $M = (\mathcal{O}_a, \ldots, \mathcal{O}_b)$ whose combined support $S_M = \bigcup_{i=a}^{b} S_i$ satisfies $|S_M| \leq 2$ and which contains at least one channel. Under those constraints, the motif map $\mathcal{E}_M = \mathcal{E}_b \circ \cdots \circ \mathcal{E}_a$ admits an exact Kraus bundle $\{K_\alpha\}_{\alpha=1}^{m}$ on the local Hilbert space $\mathbb{C}^{2^{|S_M|}}$ that satisfies

$$\sum_{\alpha=1}^{m} K_\alpha^\dagger K_\alpha = I_{2^{|S_M|}}, \qquad \mathrm{Choi}(\mathcal{E}_M) \succeq 0, \tag{2}$$

i.e. $\mathcal{E}_M$ is trace-preserving and completely positive, and may be *embedded* on the global register as $\mathcal{E}_M \otimes \mathrm{Id}_{S_M^c}$, where S_M^c is the complement of S_M. The research questions are: (**RQ1**) can $\mathcal{E}_M$ be *constructed exactly* while preserving the sequential ordering of M? (**RQ2**) does its embedded application coincide with ρ_{seq} inside a larger workload? (**RQ3**) what validation protocol must precede any acceleration claim?

Bounded support surface. Gates are restricted to U_3 and CNOT; channels to local depolarizing, amplitude damping, and phase damping with fixed rates $(p_{\mathrm{dep}}, p_{\mathrm{ad}}, p_{\mathrm{pd}}) = (0.10, 0.05, 0.07)$; and motifs to $|S_M| \leq 2$.

3 Method

3.1 Constructive Composition

The fused Kraus bundle of motif M is built incrementally in operation order (Algorithm 1). Each gate U is promoted to a single local Kraus operator and left-multiplied into the running bundle; each channel replaces the running bundle by the set of pairwise products with its Kraus operators. Because composition follows the operation order, this respects the noncommutativity of the open-system map $\mathcal{E}_M$: in general $\mathcal{E}_i \circ \mathcal{E}_j \neq \mathcal{E}_j \circ \mathcal{E}_i$.

Algorithm 1 — ComposeMotif$(M = (\mathcal{O}_a, \ldots, \mathcal{O}_b),\ S_M)$

```
d <- 2^|S_M|;   bundle <- [ I_d ]
for op in M:
    L <- LocalKraus(op, S_M)              # gate:1 op; channel: k_op ops
    bundle <- [ ell * K  for K in bundle for ell in L ]
CheckCompleteness(bundle, tol = 1e-10)    # Sum_a K_a^dag K_a = I
CheckChoiPositivity(bundle, -1e-12)       # Choi(E_M) >= 0
return bundle                             # m <= prod_i k_i operators
```

3.2 Complexity

Let $|S_M| \leq 2$, $d = 2^{|S_M|} \leq 4$, $L_M = b - a + 1$ the motif length, and $k_{\max} = \max_i k_i$ the largest single-step Kraus count ($k_i = 1$ for a gate, $k_i \in \{2, 3, 4\}$ for the three supported channels in their canonical bundles). The fused bundle has at most $m \leq k_{\max}^{L_M}$ operators; constructing it costs $O(L_M\, m\, d^3)$ (each step left-multiplies $d \times d$ matrices). Applying $\mathcal{E}_M$ to an n-qubit density matrix as a two-sided local conjugation costs $O(m \cdot 2^{2n} d)$, replacing the sequential cost $\sum_{i=a}^{b} O(k_i \cdot 2^{2n} d)$. The fused form thus trades a one-off bundle expansion for a single global apply per motif; the advantage materialises only when motif granularity or reuse dominates the expansion, which the 26-case matrix lets us assess empirically (Sec. 5).

3.3 Strict vs. Hybrid Execution

Strict execution requires *every* partition of the workload to be an eligible bounded motif; an ineligible partition raises a hard error rather than degrading silently. **Hybrid** execution routes eligible partitions through the exact channel-native path and keeps the remaining partitions – those supported by the existing fusion baseline but outside the bounded motif surface – on that baseline, recording a per-partition route label (channel-native, fused unitary island, or

supported-but-unfused). There is *no silent fallback*: a partition that neither path supports aborts the run.

4 Validation Protocol

A motif or workload counts as *validated* iff all of (i)–(v) hold, with ρ_{cn} the channel-native output and ρ_{seq} the sequential reference of Eq. (1): **(i)** Frobenius agreement $\|\rho_{\mathrm{cn}} - \rho_{\mathrm{seq}}\|_F \le 10^{-10}$; **(ii)** entrywise max-norm agreement $\|\rho_{\mathrm{cn}} - \rho_{\mathrm{seq}}\|_{\max} \le 10^{-10}$; **(iii)** trace deviation $|\mathrm{Tr}\,\rho_{\mathrm{cn}} - 1| \le 10^{-10}$ with $\rho_{\mathrm{cn}} \succeq 0$ up to the positivity floor; **(iv)** Kraus completeness residual $\|\sum_\alpha K_\alpha^\dagger K_\alpha - I\|_F \le 10^{-10}$ and Choi minimum eigenvalue $\ge -10^{-12}$; **(v)** ordered-composition guard: reversing the operation order of M shifts the output by $\gg 10^{-10}$, confirming that ordering is claim-bearing. For the external slice we additionally require agreement with an independent exact density-matrix simulator (Qiskit Aer) below 10^{-12}. The protocol is enforced by the project's automated correctness regression suite and its evidence-bundle emitters.

5 Results

5.1 Strict and Hybrid Correctness Slice

Table 1 reports the six counted cases of the bounded correctness package, five of which also carry the external cross-check. All Frobenius and max-norm residuals lie at floating-point round-off, well under the frozen 10^{-10} threshold. The two hybrid continuity anchors carry non-trivial route attribution (a mix of channel-native and baseline-routed partitions) and still reproduce the sequential reference exactly.

Table 1. Counted correctness cases: Frobenius and max-norm density differences vs. the sequential reference, trace deviation, and Frobenius difference vs. the external simulator (— when not in the external slice).

Case (counted)	Path	$\|\Delta\rho\|_F$	$\|\Delta\rho\|_{\max}$	$\|\mathrm{Tr}-1\|$	$\|\Delta\rho_{\mathrm{ext}}\|_F$
1q, U_3+local-noise chain	strict	8.0e-17	5.6e-17	2.2e-16	1.7e-15
2q, CNOT+local-noise pair	strict	2.4e-16	2.2e-16	2.2e-16	2.1e-15
2q, multi-noise entangler	strict	3.8e-16	3.3e-16	0.0	1.7e-15
2q, dense same-support motif	strict	2.6e-16	2.2e-16	0.0	1.7e-15
XXZ–HEA continuity, q=4	hybrid	1.5e-16	6.0e-17	0.0	1.8e-15
XXZ–HEA continuity, q=6	hybrid	2.0e-16	3.3e-17	2.1e-17	—

5.2 Whole-Workload Decision Matrix

The frozen 26-case matrix covers two motif-dense primary families (pair-repeat and alternating-ladder) at 8 and 10 qubits, with periodic and dense noise placements and three random seeds (24 cases), plus two sparse-noise layered nearest-neighbor control cases. Each case records a baseline trio – the sequential reference, the existing fusion baseline, and the hybrid channel-native runtime – under a median-of-three timing protocol, together with per-partition route counters. A case is labelled *channel-native-justified* iff the hybrid wall-clock speedup over the existing fusion baseline is $\ge 1.2\times$; *baseline-sufficient* iff it routes no channel-native partition; and otherwise *channel-native, not yet justified*. Table 2 gives the aggregate outcome.

Table 2. Whole-workload decision outcome over the frozen 26-case matrix.

Decision class	Cases	Criterion
Baseline-sufficient	17	no channel-native partition routed
Channel-native, not yet justified	9	routed, hybrid speedup $< 1.2\times$
Channel-native-justified	0	hybrid speedup $\ge 1.2\times$
Total	**26**	

The motif-dense families exercise the channel-native route most often, but no counted case crosses the $1.2\times$ threshold; the control cases route no channel-native partitions at all. Mathematical feasibility therefore does not, on this frozen slice, translate into a whole-workload speedup.

6 Discussion, Limitations, and Conclusion

The combined evidence supports four reusable principles. *Ordered noisy semantics* dominate the representation: Algorithm 1 carries the motif order into the bundle and guard (v) makes that order claim-bearing. A *single primary representation* (Kraus bundles) anchors the counted claim, while the Choi and Liouville forms appear only as invariant witnesses. *Physical invariant checks* (Sec. 4, (iii)–(iv)) are part of the evidence that the fused object remains a valid channel. Finally, *proof mode* and *whole-workload evaluation mode* are kept separate: the strict path closes the motif-proof obligation, while the hybrid path is the only legitimate place for whole-workload performance statements.

Three limitations remain. Mathematical feasibility and performance usefulness differ: Table 2 shows that even when the bundle is exact, the existing fusion baseline is already sufficient on $17/26$ cases and the channel-native path does not reach the $1.2\times$ threshold on the remaining 9. Dense mixed-state representation imposes a 2^{2n} memory law that no bounded fusion can relax. Broader noisy structure – correlated noise, supports $|S_M|>2$, or larger motif families – may need a different primary representation or validation rule. The right take-away is therefore *not* "noisy fusion is solved," but "a bounded exact path exists, with an explicit validation protocol and an open performance question."

Reproducibility. All values in Tables 1–2 are produced by the correctness and performance validation pipelines shipped with `SQUANDER` [1]. The frozen numerical policy is 10^{-10} for the density-agreement and Kraus-completeness tolerances and -10^{-12} for the Choi positivity floor; the fixed noise rates are $(p_{\mathrm{dep}}, p_{\mathrm{ad}}, p_{\mathrm{pd}})=(0.10, 0.05, 0.07)$; the structured seeds are $\{20260318, 20260319, 20260320\}$; and the channel-native-justified threshold is a $1.2\times$ hybrid speedup over the existing fusion baseline. The reported results were obtained at source revision `8a3d2b39` (tag `1.9.6-146-g8a3d2b39`) under Python 3.13 in the project's documented build environment.

References

[1] SQUANDER, source revision `8a3d2b39`, `github.com/rakytap/sequential-quantum-gate-decomposer`.

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Univ. Press (2010).

[3] C. J. Wood, J. D. Biamonte, and D. G. Cory, *Tensor networks and graphical calculus for open quantum systems*, Quantum Inf. Comput. 15, 759–811 (2015).

[4] A. Li, O. Subasi, X. Yang, and S. Krishnamoorthy, *Density Matrix Quantum Circuit Simulation via the BSP Machine on Modern GPU Clusters*, SC20.

[5] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, *QuEST and High Performance Simulation of Quantum Computers*, Sci. Rep. 9, 10736 (2019).

[6] Y. Suzuki et al., *Qulacs: a fast and versatile quantum circuit simulator for research purpose*, Quantum 5, 559 (2021).

[7] J. Clark, T. S. Humble, and H. Thapliyal, *TDAG: Tree-based DAG Partitioning for Quantum Circuits*, ACM GLSVLSI 2023.

[8] J. Clark, T. S. Humble, and H. Thapliyal, *GTQCP: Greedy Topology-Aware Quantum Circuit Partitioning*, arXiv:2410.02901.

[9] X.-C. Wu, M. G. Davis, F. T. Chong, and C. Iancu, *QGo: Scalable Quantum Circuit Optimization Using Automated Synthesis*, arXiv:2012.09835.

[10] L. Xu, E. H.-M. Sha, Y. Song, and Q. Zhu, *QMin: Quantum Circuit Minimization via Gate Fusions for Efficient State Vector Simulation*, Quantum Inf. Process. 25, 6 (2026).

From Centralized to Distributed Digital Twins: A Review

Pál Mihály

Abstract: Distributed computing systems built on Fog and Edge Computing paradigms, combined with the Internet of Things, have introduced new challenges in latency, scalability, and system reliability. Digital Twin technology has emerged as a promising approach to manage the growing complexity of such environments, however traditional centralized DT architectures suffer from critical limitations including scalability bottlenecks, excessive network load, centralized control overhead, and single points of failure. This paper reviews existing DT solutions for distributed Fog-Edge-IoT environments, examining how these limitations are addressed through Distributed Digital Twin approaches. This paper also identified open research challenges in architectural decomposition, consensus management, heterogeneous resource handling, and data model lifecycle management.

Keywords: Digital Twin, Distributed Systems, Fog Computing, Edge Computing

1 Introduction

The emergence of distributed systems has introduced new challenges and concepts, including paradigms such as Fog and Edge Computing, as well as the Internet of Things (IoT) that refers to miniaturized interconnected devices generating and exchanging data. To provide low response times for real time applications, the data originated from IoT sensors are processed closer to the end users on resource constrained fog and edge nodes. The main idea behind such distributed computing nodes is to reduce the time it takes for data to reach the processing node. Since there are numerous layers and devices on the network through which data must pass, if we reduce the digital distance that data must travel from creation to processing, then latency will also decrease. Another advantage of Fog and Edge Computing is the privacy of data achieved by keeping the information close to their origin, on edge devices or near intermediary fog nodes, unlike when all processing takes place on a server, typically in a cloud that is digitally distant from our data [1].

To deal with ever-increasing complexity of Fog-Edge-IoT systems, and to provide reliably execution environments for IoT applications, the concept of Digital Twin (DT) has been introduced. The concept of Digital Twin has its roots in product lifecycle management and has since evolved into a foundational paradigm of cyber-physical integration [2]. A DT allows the creation of a dynamic and virtual representation of a physical object, process, or system with continuously synchronized data between the real-world counterpart. In typical use-case the DT comprises three core components which are the physical entity, its virtual representation, and the data connections that keep the two aligned in real time. An up-to-date DT that continuously ingests sensor data and updating its internal state model representing the main characteristics of the physical counterpart, enables a broad range of capabilities. With a system-wide DT, we can perform real-time monitoring, anomaly detection, autonomous control, and predictive maintenance as well. These functionalities make the DT concept a robust and powerful tool for managing complex distributed systems.

In this paper, we provide an overview of how DT solutions are integrated into distributed systems, with a primary focus on Fog and Edge Computing, and their role in enabling distributed Digital Twin architectures.

2 Existing Digital Twin Solutions for Distributed Systems

Traditional DT use cases include smart manufacturing optimization, asset maintenance (e.g., aircraft engine and wind turbine), and smart city modeling. However, shifting towards distributed

applications, several challenges may arise, including scalability, centralized control, increased network traffic and latency, and single point of failure.

When a centralized DT service is responsible for maintaining comprehensive digital representations of an increasing number of system components, scalability challenges emerge as it must handle a growing number of simultaneous data streams and state updates. This quickly becomes a bottleneck: a single service absorbs the computational, memory, and storage demands of thousands or millions of active DTs. Although central control has advantages on a small scale, when all the management logic is concentrated in one place, every decision must pass through that central point. This will cause coordination delays and make the system slow to react to local events. In real-time and safety-critical environments, relying on a remote server to process data and respond to time-critical events introduces unacceptable delays in decision making.

The single point of failure can easily degrade system performance. In case of a hardware fault, software crash, cyber-attack, or network outage, every twin it hosts becomes unavailable simultaneously. All capabilities such as monitoring, analytics, and control functions across connected physical assets are disrupted at once. This total dependency on a single infrastructure point is just incompatible with the high-availability requirements of industrial, healthcare, or smart city deployments.

To solve the aforementioned problems with centralized DT services, Distributed Digital Twin (DDT) approaches have been explored in prior work. In this work, we adopt this concept from a different perspective, focusing on its application in Fog-Edge-IoT environments.

Instead of maintaining a single virtual replica in the cloud, a DDT decomposes twin functionalities across multiple nodes distributed along fog and edge nodes, placing DT logic closer to where the data are generated. Similar to the benefits observed in Fog-Edge-IoT systems, this approach also enables lower latency in accessing and processing DT-related information. In other words, each physical entity is in connection with a lightweight twin instance deployed on a nearby edge or fog node, enabling time-critical synchronization locally, while heavier analytics and aggregation tasks are offloaded to higher-tier nodes as needed. The horizontal distribution improves scalability, and local autonomy reduces central control overhead. The proximity to data sources minimizes network load and latency, and due to the distribution, there is no single central dependency, eliminating the risk of system-wide failure. Nevertheless, the architecture of the DDT concept introduces some new challenges, such as twin-to-twin communication, state consistency, and lifecycle management across nodes.

The synchronization modes of distributed DT instances can vary depending on their usage. It could be real-time, if the immediate anomaly detection and real-time monitoring are important. It could also be periodic following a recurring pattern or event-driven, which will decrease network traffic because it is synchronized with the physical counterpart when something happens.

Edge-layer twins support time-critical decisions by reacting to sensor readings locally. The key advantage of twins in the edge layer is the ultra-low latency, but the edge nodes have limitations caused by the amount of data provided to them (i.e., they only see their own local data), also their computing capacity that prevent the execution of resource-heavy algorithms.

Resource allocation is also important, since not all twins require the same amount of resources, as their computational needs may vary. The allocation strategy determines how much CPU, memory, bandwidth, and storage should be assigned to each twin instance, ideally adapting dynamically to changing workloads. Since nodes can fail, the system must be designed to survive partial failures, ensuring fault tolerance. This requires techniques such as the creation of twin replicas, checkpoints (periodically saving the twin state), and failover mechanisms that automatically migrate a twin to a healthy node when its host fails.

In terms of architectural design, two main patterns can be identified in DDTs. The first is the System-of-Systems (SoS), and the other one is the Hierarchical. In the SoS model, the DDT is made up of multiple independent twin subsystems, each managing a distinct physical domain,

that interact through well-defined interfaces. In addition, no single twin has global authority. The whole system behavior emerges from peer-to-peer or loosely coupled interactions. This pattern is common in scenarios like smart cities or supply chain, where many autonomous stakeholders must collaborate without centralized coordination. In contrast, in the Hierarchical model, twins are organized in a tree-like structure that mirrors the Edge-Fog-Cloud hierarchy. The lower-level twins manage individual assets or devices, the mid-level twins coordinate and aggregate groups of lower twins, and in the top-level twins provide a global view of the entire system. The Hierarchical pattern is well-suited to smart factories, infrastructure monitoring, and industrial automation where there is a natural organizational hierarchy.

Table 1: Comparison of reviewed Digital Twin solutions according to architecture, deployment layer, twin granularity, latency reduction, privacy/security aspects, and application domain

Paper	Paper Type	DT Architecture	Deployment Layer	Twin Granularity	Latency Reduction	Privacy / Security	Application Domain
[2]	Literature Review	Hierarchical (Unit ->System ->SoS)	Cloud	Unit / System / SoS level	N/A	Identified as challenge	Manufacturing, Aerospace, Healthcare, Smart Cities, Agriculture, Automotive
[3]	Survey	Three-component: physical products, virtual models, data interface	Cloud + Fog + Edge	Product / Production / Performance	Identified as future need	N/A	Manufacturing, Healthcare, Smart Cities, Agriculture
[1]	Experimental	Microservice based (Cloud + Fog)	Fog + Cloud	Device / Sensor level	Proven: -54% (Fog + Cloud), -64% (Fog only)	N/A	IoT, Smart Industry
[4]	Conceptual + Experimental	Five-layer (Physical Space, Communication, Data Preprocessing, Twin, Services)	Physical + Edge + Cloud (DDT) / Physical only (LDT)	Per physical asset, split by complexity	Core design goal of LDT and DDT	Local data processing (LDT)	Smart Industry, Healthcare, Smart Cities
[5]	Systematic Review	Domain-specific, no unified architecture	Mostly Cloud	Varies by domain and maturity	Identified as challenge only	Identified as challenge only	Smart Cities, Logistics, Medicine, Engineering, Automotive
[6]	Framework + Experimental	ISO 23247-based distributed framework with microservices and containerization (Docker)	Fog + Cloud	Component-level	Identified as metric, not experimentally evaluated	N/A	IIoT, Predictive Maintenance, Wind Energy

Table 1 summarizes the reviewed DT solutions from a distributed systems perspective. The comparison shows that most approaches address distributed deployment, latency, or architectural decomposition, but none of them provide a complete solution for scalable, consistent, and heterogeneous Distributed Digital Twin environments.

3 Open Challenges and Research Directions

The review of existing DT solutions for distributed systems reveals that, despite significant progress, there are still open challenges to be addressed, such as architectural design, consistency management, and the lifecycle of the data model.

One of the challenges in DDT design is determining how to decompose a monolithic DT into multiple cooperating distributed instances without losing the coherent system-wide view that a centralized twin can provide. As we can see in Table 1, existing works address this differently. In [4] they decompose twin functionality across physical, edge, and cloud layers by task complexity, while in [6] they adopt an ISO 23247-based microservice architecture to achieve modular decomposition. However, a unified decomposition methodology has not yet been established. Another challenge is to choose the appropriate level of twin granularity. First, we should choose the level at which we want to operate with our twins. If we operate at the level of individual devices or microservices, then we use fine-grained twins that offer higher precision and faster local reaction times, but have a significant communication and management overhead as the number of twin instances increases. In contrast, coarse-grained twins operating at the subsystem or layer level, and as it is shown in Table 1, granularity choices are very widely across the reviewed works, from the device and sensor level in [1] to the component level in [6] and the SoS level in [2].

When we distribute twin functionality across multiple nodes, then the maintaining of a consistent and up-to-date system representation becomes non-trivial. All twin instances hold a

partial view of the system, ensuring that these views remain coherent, especially in the presence of node failures, concurrent updates, or network delays. Although synchronization strategies such as event-driven, real-time, and periodic modes are discussed in the literature, there is no widely adopted general consensus protocol specifically designed for DDT environments. Established distributed systems mechanisms such as Paxos and Raft for agreement, CRDTs for conflict-free replicated state management, and gossip protocols for scalable information dissemination may provide useful building blocks for maintaining consistency across distributed twin instances. Strong consistency can conflict with the low-latency requirements of real-time IoT applications. Synchronizing all twins after every state change introduces communication overhead, while relaxed consistency may lead to outdated or conflicting system representations. This consistency-latency trade-off remains unresolved in the reviewed works.

Edge and fog environments are inherently heterogeneous, consisting of nodes with widely varying storage, memory, processing capacity, and network connectivity. In contrast, cloud infrastructure offers relatively uniform and elastic resources, edge and fog nodes are often constrained and diverse. This makes it difficult to deploy and manage twin instances uniformly. If a strategy is suitable for a powerful fog server, that does not mean it will be suitable on a resource-constrained edge device. Training machine learning models or running physics-based simulations on edge devices is often not practical, yet offloading these tasks entirely to the cloud will cause latency problems again that DDT architectures aim to solve.

As each twin instance in a DDT maintains a local model trained on its own data sources, integrating these local models into a globally consistent and accurate representation becomes a significant challenge. Federated Learning (FL) offers a promising approach by allowing model updates to be aggregated without centralizing raw data. This also preserves data privacy, a concern identified in multiple reviewed works but concretely addressed only in [4] through local data processing. In dynamic IoT environments, the operational conditions of physical assets change over time, thus deciding when to retrain or update a model, without disrupting the real-time operation of the twin, is a challenge that none of the reviewed works fully resolves.

References

[1] Francisco Paiva Knebel, Juliano Araujo Wickboldt, Edison Pignaton de Freitas. A Cloud-Fog Computing Architecture for Real-Time Digital Twins. [Submitted on 11 Dec 2020 (v1), last revised 7 May 2021 (this version, v3)]. https://doi.org/10.48550/arXiv.2012.06118

[2] Singh M, Fuenmayor E, Hinchy EP, Qiao Y, Murray N, Devine D. Digital Twin: Origin to Future. Applied System Innovation. 2021; 4(2):36. https://doi.org/10.3390/asi4020036

[3] Zhang, R., Wang, F., Cai, J. et al. Digital twin and its applications: A survey. Int J Adv Manuf Technol 123, 4123-4136 (2022). https://doi.org/10.1007/s00170-022-10445-3

[4] S. Rauf, F. Muhammad, A. Badshah, H. Alasmary, M. Waqas and S. Chen, "Novel Digital Twin Deployment Approaches: Local and Distributed Digital Twin," in IEEE Access, vol. 13, pp. 72142-72152, 2025, https://doi.org/10.1109/ACCESS.2025.3561354

[5] Botin-Sanabria DM, Mihaita A-S, Peimbert-Garcia RE, Ramirez-Moreno MA, Ramirez-Mendoza RA, Lozoya-Santos JdJ. Digital Twin Technology Challenges and Applications: A Comprehensive Review. Remote Sensing. 2022; 14(6):1335. https://doi.org/10.3390/rs14061335

[6] Abdullahi I, Longo S, Samie M. Towards a Distributed Digital Twin Framework for Predictive Maintenance in Industrial Internet of Things (IIoT). Sensors. 2024; 24(8):2663. https://doi.org/10.3390/s24082663

List of Authors

Abdel-Fattah, Sami: University of Szeged, Hungary,
E-mail: khalid@inf.u-szeged.hu

Alharan, Abbas: University of Pannonia, Hungary,
E-mail: abbas.alharan@phd.mik.uni-pannon.hu

Al-Kharsan, Hiba Adil: University of Szeged, Hungary,
E-mail: hibaadil@inf.u-szeged.hu

Antal, Gábor: University of Szeged, Hungary,
E-mail: antal@inf.u-szeged.hu

Balogh, Etele: University of Szeged, Hungary,
E-mail: baloghe@inf.u-szeged.hu

Bánhelyi, Balázs: University of Szeged, Hungary,
E-mail: banhelyi@inf.u-szeged.hu

Benaissa, Moenes: University of Miskolc, Hungary,
E-mail: moenesb7@gmail.com

Bertók, Botond: University of Pannonia, Hungary,
E-mail: bertok.botond@mik.uni-pannon.hu

Brand, Ádám: University of Pannonia, Hungary,
E-mail: brand.adam@mik.uni-pannon.hu

Buttyán, Levente: Budapest University of Technology and Economics, Hungary,
E-mail: buttyan@crysys.hu

Delavari, Zahra: University of Szeged, Hungary,
E-mail: delavari@inf.u-szeged.hu

Domonkos, Ádám: Budapest University of Technology and Economics, Hungary,
E-mail: domonkosadam01@gmail.com

Ekler, Péter: Budapest University of Technology and Economics, Hungary,
E-mail: peter.ekler@aut.bme.hu

Forstner, Bertalan: Budapest University of Technology and Economics, Hungary,
E-mail: Forstner.Bertalan@aut.bme.hu

Földvári, András: Budapest University of Technology and Economics, Hungary,
E-mail: foldvari.andras@vik.bme.hu

Frits, Márton: University of Pannonia, Hungary,
E-mail: frits.marton@mik.uni-pannon.hu

Guzsvinecz, Tibor: University of Pannonia, Hungary,
E-mail: guzsvinecz.tibor@zek.uni-pannon.hu

Gyarmati, László: University of Pannonia, Hungary,
E-mail: gyarmati.laszlo@phd.mik.uni-pannon.hu

Hajnal, Péter: University of Szeged, Hungary,
E-mail: hajnalp@inf.u-szeged.hu

Hegedűs, Péter: University of Szeged, Hungary,
E-mail: hpeter@inf.u-szeged.hu

Horváth, Anna Lili: Eötvös Loránd University, Hungary,
E-mail: horvathlili237@gmail.com

Huszti, Andrea: University of Debrecen, Hungary,
E-mail: huszti.andrea@inf.unideb.hu

Jász, Judit: University of Szeged, Hungary,
E-mail: jasy@inf.u-szeged.hu

Juhász, Zoltán: University of Pannonia, Hungary,
E-mail: juhasz.zoltan@mik.uni-pannon.hu

Karikó, Csongor Csanád: Eötvös Loránd University, Hungary,
E-mail: kcscs@inf.elte.hu

Kégli, Zoltán: Eötvös Loránd University, Hungary,
E-mail: kegli.zoltan@gmail.com

Kicsi, András: University of Szeged, Hungary,
E-mail: akicsi@inf.u-szeged.hu

Kolláth, István: University of Szeged, Hungary,
E-mail: kollath@inf.u-szeged.hu

Kósa, István: University of Szeged, Hungary,
E-mail: kosa.istvan@med.u-szeged.hu

Kotroczó, Roland: Eötvös Loránd University, Hungary,
E-mail: kotroczo.roland@inf.elte.hu

Kovács, Bence: Budapest University of Technology and Economics, Hungary,
E-mail: kovacsbence.se@gmail.com

Kovács, Bertalan: University of Pannonia, Hungary,
E-mail: nemaberci@gmail.com

Kovács, Dániel: Budapest University of Technology and Economics, Hungary,
E-mail: daniel.kovacs.1@edu.bme.hu

Kovács, László József: University of Miskolc, Hungary,
E-mail: laszlo.kovacs@uni-miskolc.hu

Kozsik, Tamás: Eötvös Loránd University, Hungary,
E-mail: kto@inf.elte.hu

Magyar, Attila: University of Pannonia, Hungary,
E-mail: magyar.attila@mik.uni-pannon.hu

Maliga, Dávid: Budapest University of Technology and Economics, Hungary,
E-mail: dmaliga@crysys.hu

Mihály, Pál: University of Szeged, Hungary,
E-mail: mihalypal@inf.u-szeged.hu

Mondok, Milán: Budapest University of Technology and Economics, Hungary,
E-mail: mondokm@edu.bme.hu

Nagy, Csaba: University of Debrecen, Hungary,
E-mail: nagy.csaba@inf.unideb.hu

Nagy, Noel: University of Szeged, Hungary,
E-mail: nagynoel@inf.u-szeged.hu

Nagy, Róbert: University of Pannonia, Hungary,
E-mail: nagy.robert@mik.uni-pannon.hu

Namrouti, Hanan: University of Pannonia, Hungary,
E-mail: hanan.namrouti@phd.mik.uni-pannon.hu

Oláh, Norbert: University of Debrecen, Hungary,
E-mail: olah.norbert@inf.unideb.hu

Palágyi, Kálmán: University of Szeged, Hungary,
E-mail: palagyi@inf.u-szeged.hu

Pap, Zsigmond: Ericsson,
E-mail: zsigmond.pap@ericsson.com

Papp, Gergő: Eötvös Loránd University, Hungary,
E-mail: gergo.pappbalint@gmail.com

Pataricza, András: Budapest University of Technology and Economics, Hungary,
E-mail: pataricza.andras@vik.bme.hu

Pengő, Edit: University of Szeged, Hungary,
E-mail: pengoe@inf.u-szeged.hu

Pomázi, Krisztián: Budapest University of Technology and Economics, Hungary,
E-mail: pomazi.krisztian@aut.bme.hu

Rajkó, Róbert: Óbuda University, Hungary,
E-mail: rajko.robert@uni-obuda.hu

Rakyta, Péter: Eötvös Loránd University, Hungary,
E-mail: peter.rakyta@ttk.elte.hu

Samedov, Aleksandr: Eötvös Loránd University, Hungary,
E-mail: emsmx4@inf.elte.hu

Sándor, József: Budapest University of Technology and Economics, Hungary,
E-mail: jsandor@crysys.hu

Sik-Lányi, Cecília: University of Pannonia, Hungary,
E-mail: lanyi.cecilia@mik.uni-pannon.hu

Siket, István: University of Szeged, Hungary,
E-mail: siket@inf.u-szeged.hu

Somfai, Ellák: Eötvös Loránd University, Hungary,
E-mail: somfaiellak@inf.elte.hu

Sulaymonov, Yusufbek: University of Pannonia, Hungary,
E-mail: yusufbek.sulaymonov@phd.mik.uni-pannon.hu

Szálka, Brigitta: University of Pannonia, Hungary,
E-mail: szalka.brigitta@gmail.com

Szalai-Gindl, János Márk: Eötvös Loránd University, Hungary,
E-mail: szalaigindl@inf.elte.hu

Szekeres, Krisztián: OX-IT,
E-mail: krisztian.szekeres@oxit.hu

Szolnoki, Norbert Sándor: University of Szeged, Hungary,
E-mail: szolnoki@inf.u-szeged.hu

Szűcs, Veronika: University of Pannonia, Hungary,
E-mail: szucs.veronika@mik.uni-pannon.hu

Toldi, Balázs Ádám: Budapest University of Technology and Economics, Hungary,
E-mail: balazs.toldi@edu.bme.hu

Tóth, Bálint: University of Pannonia, Hungary,
E-mail: toth.balint1225@gmail.com

Vad, Viktor: Eötvös Loránd University, Hungary,
E-mail: vad@inf.elte.hu

Valasek, Gábor: Eötvös Loránd University, Hungary,
E-mail: valasek@inf.elte.hu

Vassányi, István: University of Pannonia, Hungary,
E-mail: vassanyi@almos.vein.hu

Világos, Emil: Neumann János University, Hungary,
E-mail: emil.vilagos@oxit.hu

Vincze, Nándor Ákos: University of Szeged, Hungary,
E-mail: nvincze@inf.u-szeged.hu

Vörös, Vivien: University of Szeged, Hungary,
E-mail: vivien19@inf.u-szeged.hu

Notes