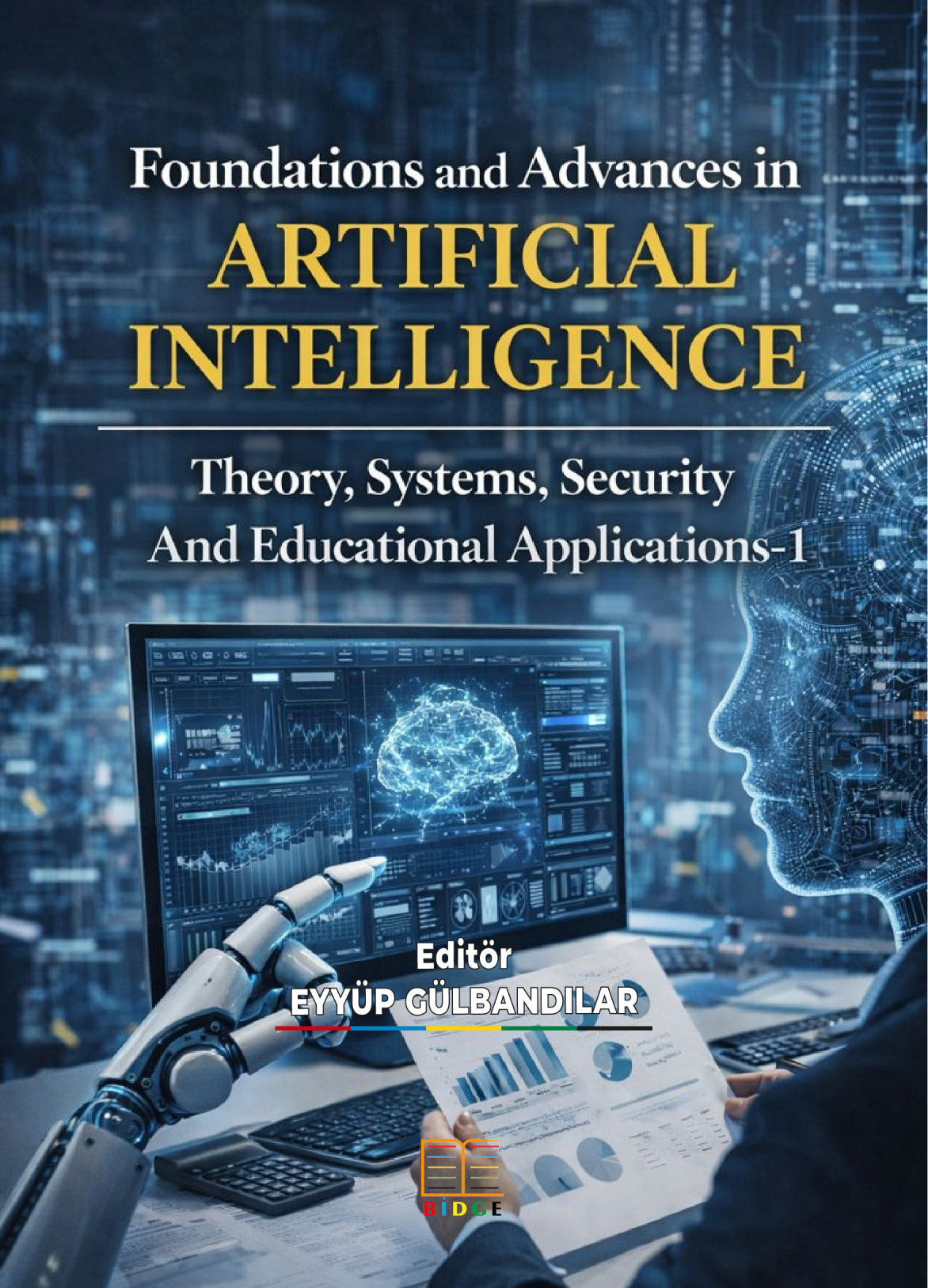
Foundations and Advances in ARTIFICIAL INTELLIGENCE

Theory, Systems, Security And Educational Applications-1



Editör
EYYÜP GÜLBANDILAR

BİDGE

BİDGE Yayınları

Foundations and Advances in Artificial Intelligence: Theory, Systems, Security And Educational Applications-1

Editör: EYYÜP GÜLBANDILAR

ISBN: 978-625-372-946-2

1. Baskı
Sayfa Düzeni: Gözde YÜCEL
Yayınlama Tarihi: 2025-12-25
BİDGE Yayınları

Sertifika No: 71374
Yayın hakları © BİDGE Yayınları
www.bidgeyayinlari.com.tr - bidgeyayinlari@gmail.com
Krc Bilişim Ticaret ve Organizasyon Ltd. Şti.
Güzeltepe Mahallesi Abidin Daver Sokak Sefer Apartmanı No: 7/9 Çankaya / Ankara

PREFACE

The trajectory of Artificial Intelligence (AI) has shifted decisively from a niche theoretical pursuit to the very backbone of contemporary computing, cybersecurity, and algorithmic decision-making. We are currently witnessing a period where machine learning architectures and system-level refinements do not just process data—they redefine how we secure infrastructures and interpret complex information. This book emerged from a necessity to document these rapid shifts through a lens that balances technical rigor with practical interpretability.

Rather than offering a mere compilation of studies, this book orchestrates a dialogue between foundational theory and empirical validation. We begin by examining the subtle evolution of covert channel detection, moving progressively toward the systemic nuances that dictate AI model performance. A significant portion of our inquiry is dedicated to the "black box" of deep learning; by dissecting activation functions and word embedding methodologies, we aim to expose the mathematical constraints that often go unaddressed in standard literature.

Crucially, the rise of Explainable AI (XAI) is treated here not as a luxury, but as a prerequisite for security-sensitive deployments. Our focus on phishing detection underscores a vital argument: for AI to be truly effective in real-world security, it must be as transparent as it is powerful.

What perhaps distinguishes this work most is its forward-looking exploration of the academic landscape itself. The proposal of a fully AI-integrated university learning ecosystem—specifically tailored for graduate-level engineering and educational sciences—challenges the traditional boundaries of pedagogy. It positions AI not merely as a supportive tool, but as a catalyst for a systemic redesign of how knowledge is produced and shared.

This collection is designed for those navigating the intersection of research and application—be they graduate students, scholars, or practitioners. It is our hope that these pages do more than inform; we intend for them to provoke new questions and inspire the next wave of interdisciplinary innovation in the ever-evolving AI domain.

Dr.Eyyup GULBANDILAR

Editor

İÇİNDEKİLER

BÖLÜM 1

Explainable AI for Phishing Detection: Trends, Challenges, and Future Directions

ÖZGÜR TONKAL[1]
HALİM ASLIYÜKSEK[2]

Introduction

The continuously escalating and increasingly sophisticated cybersecurity threats are making phishing attacks one of the most significant, harmful, and enduring problems of the modern digital age. Phishing attempts continue to circumvent traditional signature-based security measures through impersonating legitimate institutions with the intent to acquire sensitive information by illegal means (Yakandawala, Madushanka, & Ilmini, 2024). The threat's reach is no longer ignorable. According to a report published in 2024, AI-powered social engineering methods were identified as being used in more than 932,923 incidents in the third quarter of 2024 (APWG, 2025).This dramatic increase highlights the need for adaptable and effective defence systems. Implementing AI-powered countermeasures is considered essential for interpreting these

[1]Assistant Professor, Department of Software Engineering, Samsun University, Orcid: 0000-0001-7219-9053
[2] Master's Student, Department of Software Engineering, Samsun University, Orcid: 0009-0003-1049-7786

attacks, which employ complex scenarios, and for generating rapid responses.

Efforts have been made to develop effective methods to counter the increasing volume of attacks and their complex methods. In this direction, advanced artificial intelligence and machine learning models have been developed, including classifiers such as SVM and RF, which offer high detection performance. For example, studies conducted on the detection of phishing emails have found that certain M5 models can perform well, and also that the SVM model achieved 96% accuracy in tests (Yakandawala, Madushanka, & Ilmini, 2024). However, due to the "black box" nature of complex algorithms, these high-performance systems will often limit their effectiveness. These types of systems may classify a URL or email as malicious, but do not provide a plausible rationale for the decision. This lack of transparency undermines the integrity of the system and erodes user trust. Failure to provide justification has increased the success rate of phishing attacks by causing users to be unable to understand the risk and/or not trust the defence system. Lack of interpretability hinders operational feedback, verifiability and overall acceptance of MO-based solutions in critical security environments.

This critical gap is the field that Explainable Artificial Intelligence (XAI) is addressing. XAI, the technical models of machine learning, are examined in detail and made transparently accessible to security analysts and general users. The integration of methods such as LIME (Locally Interpretable Model-Agnostic Explanations) and the use of intrinsically interpretable classifiers such as Explainable Boosting Machine (EBM), provides clear and user-friendly comments for XAI model predictions (Khaleel & Salihi, 2019). The function of rules and rationales in the explanatory security mechanism is not only to foster trust and user awareness, but it also shapes compliance with regulations and facilitates the

purpose of trustworthy AI in advanced infrastructure, such as secure IoT and Cloud-based Cyber-Physical Systems (CPS).

This section outlines how XAI approaches are utilised in phishing detection systems and explains their operation within these systems. In particular, reducing the uncertainty caused by black-box models and increasing confidence in security analysts' decisions is crucial. To this end, this study is not merely a review of current XAI applications and technology trends. It also aims to comprehensively address the practical challenges of building a scalable and reliable cybersecurity architecture, as well as future approaches.

The remainder of this section is organized as follows: Section 2 provides a theoretical foundation by presenting a literature review of specific applications of XAI techniques (LIME, SHAP, EBM) used in the classification and detection of phishing attacks. This fundamental understanding will enable a more critical assessment of how complex systems are replacing traditional models, as discussed in subsequent sections. Consequently, the latest trends in the field, including hybrid architectures and optimisation-based approaches, are discussed in Section 3. The accuracy-interpretability linkage and scalability are then addressed as significant technical and ethical challenges. Section 5 describes promising directions for future research, including multimodal analysis and human-centred XAI systems. Finally, the results are summarised in Section 6, which elaborates on the importance of XAI in developing future phishing defence mechanisms.

Literature Review on Explainable AI in Phishing Detection

This section provides a comprehensive review of existing literature to demonstrate the specific role of explainable artificial intelligence (XAI) in phishing detection. This analysis is structured the same way the technology has evolved in the field. The analogy is that a review of detection technologies ranges from first-

generation heuristic-based systems to later high-performance models that typically operate as opaque black boxes. Thus, in short, it provides a review of how the models' increased accuracy became an issue of transparency.

The absence of a way to interpret such complex decision pathways will lead to a lack of trust in the operationalisation of these models and inhibit real-time threat response. After that, basic XAI techniques are explained, and the growing literature applying these techniques towards phishing is critically assessed.

An Evolutionary Overview of Phishing Detection Paradigms

Phishing detection methods have undergone various phases in their evolution, as each new generation was intended to address the limitations of its predecessor. Although this trend has markedly improved predictive accuracy, the problem of model interpretability has also intensively become worse. As deep learning architectures become more prevalent, the finer details of how they classify outcomes become very vague. This lack of specificity poses a major hurdle for forensic analysis and threat validation. Hence, we are facing the ‘black box’ dilemma, which makes the integration of XAI frameworks essential rather than optional.

The first defence mechanisms were based mostly on rule systems and heuristics. More specifically, early approaches relied on manually maintaining blacklists of known malicious domains and applying static, rule-based procedures to identify anomalous patterns within URLs and email content. These methods yielded notable outcomes, including the detection of raw IP addresses embedded in hyperlinks and the identification of common social engineering keywords (Whittaker, Ryner & Nazif, 2010). Although computationally efficient and straightforward to implement, their effectiveness was inherently limited by their static design. Inputs that deviated from predefined patterns—whether through URL-

shortening services, obfuscation techniques, or homograph attacks—could easily circumvent such defenses. Moreover, it became increasingly evident that adversarial techniques were rapidly advancing. Threat actors frequently register new domains or alter content structures to evade detection, necessitating continuous manual updates and leading to elevated false positive rates. The operational burden and brittleness of heuristic-based frameworks thus became unsustainable.

These constraints prompted a transition toward traditional machine learning methodologies. In this phase, static rules were supplanted by data-driven statistical models. Rich feature sets were engineered from URL structures and email content, and algorithms such as Support Vector Machines (SVM), Naive Bayes (NB), and Random Forests (RF) were employed to perform threat classification. Sahingoz et al. (2019) is an example of this. The seven ML architectures they trained were solely based on attributes from the URL. A random forest model was shown to achieve an accuracy exceeding 97.8% which proves the effectiveness of this method (Sahingoz, Buber, Demir, & Diri, 2019). The accuracy in categorization was so high that one could observe a statistical discrimination overthrowing a rigid rule-matching if a certain data representation was good enough. Performance certainly witnessed large strides in the ability to detect complex non-linear patterns, but remained very much dependent on the quality and relevance of human-engineered features. This period's representative studies are summarized overall in Table 1.

Table 1. Some studies have been conducted using classical machine learning methods

Author Names	**Methods**	**Performance Metrics**
Khaleel & Salihi, 2019	NB, IBK, RF, J48	Accuracy: 92.95% Sensitivity: 92% F1 Score: 89%

Dedeturk & Akay (2020)	LR + Artificial Bee Colony and Classical Machine Learning Methods	Accuracy: 93.2%
Junnarkar, Adhikari, Fagania, Chimurkar, & Karia, 2021	SVM, NB, K-Nearest Neighbors (KNN), RF, DT, KNN	Accuracy: 97.83%
Rayan, 2022	RF, DT	Accuracy: 95%
Alsuwit, Haq, & Aleisa, 2024	LR, NB, RF, Artificial Neural Network (ANN)	Accuracy: 98% F1 Score: 97.5%

Kaynak: Asliyuksek, Tonkal & Kocaoglu, 2025.

The use of Deep Learning (DL) architectures and Transformers has led to an enhanced detection system or method. The use of Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and most importantly, Bidirectional Encoder Representations from Transformers (BERT) significantly improved the detection capabilities. These models are characterized by their ability to extract relevant and high-level features from unstructured text. As a result, manual feature engineering was less dependent on labour. Furthermore, these systems are capable of capturing subtle contextual and semantic cues that earlier methodologies routinely overlooked. In complex spear-phishing scenarios—where the effectiveness of deception relies more on narrative plausibility than on overt structural irregularities—this deep semantic understanding becomes a decisive advantage. A representative example is the Phish-BERT framework proposed by Wang et al., which demonstrates superior performance compared to both classical machine learning approaches and generic deep learning baselines in detecting highly sophisticated semantic attacks (Wang, Zhu, Xu, Qin, Ren, & Ma, 2023).

As summarized in Table 2, although these models have established new state-of-the-art accuracy levels, they also introduce substantially greater architectural complexity, often comprising millions of parameters. Consequently, their internal reasoning

processes remain opaque, reinforcing long-standing concerns regarding interpretability and transparency.

Table 2. Representative Studies Using Deep Learning and Transformers for Phishing Detection

Author Names	Methods	Performance Metrics
Tida & Hsu, 2022	BERT + Machine Learning	Accuracy: 97%
Nasreen, Khan, Younus, Zafar, & Hanif, 2024	CNN + Novel feature selection	Accuracy: 99.14%
Meléndez, Ptaszynski, & Masui, 2024	RoBERTa + SVM	F1 Score: 99.51% Accuracy: 99.43%
Chandan, Dsouza, George, & Bhadra, 2023	BERT + ML Algorithms	Accuracy: 98%
AbdulNabi & Yaseen, 2021	BERT	F1 Score: 98.66% Accuracy: 98.67%

This top performance definitely solved the "black box" problem. When a model flags an email with 99.8% certainty but provides no explanation that a human can understand, a security analyst cannot verify the alert or comprehend the nature of the attack. This is an operational problem. The overall progression of these technological paradigms, shaped by their inherent rules and constraints, is synthesized in Table 3. The persistent trade-off between performance and transparency represents the fundamental gap that Explainable AI (XAI) seeks to bridge.

Table 3. Comparative Evolution of Phishing Detection Approaches

Criterion	Paradigm 1: Heuristics & Signatures	Paradigm 2: Classical Machine Learning	Paradigm 3: Deep Learning / Black-Box AI
Core Principle	Relying on static, human-defined rules and signatures.	Learning statistical patterns from data through feature engineering.	Automatically learning hierarchical and complex representations from raw data.

Example Techniques	Blacklists, keyword matching, heuristic analysis of URL structure.	Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB).	Convolutional Neural Networks (CNN), LSTM, Transformer Models (e.g., BERT).
Strengths	Simple to understand and implement, it provides fast protection against known threats.	Higher accuracy, better adaptation to novel attack patterns.	State-of-the-art accuracy and performance, eliminates need for manual feature engineering.
Weaknesses / Limitations	Easily bypassed, ineffective against zero-day attacks, high false-positive rates.	Requires extensive and brittle feature engineering, fails to capture contextual meaning.	"The Black-Box Problem": Decisions are not transparent, lack of interpretability, high computational cost.

Table 3 provides an overview of the evolution of phishing detection approaches, tracing the progression from heuristic techniques to classical machine learning and, subsequently, to deep learning–based models. Although each paradigm introduces notable gains in accuracy, adaptability, and automation, it also brings distinct constraints related to explainability, transparency, and computational cost. This comparison highlights the inherent tension between performance and interpretability, underscoring why XAI methods have become a crucial component of modern detection architectures.

The Emergence and Application of XAI in Phishing Detection

Explainable Artificial Intelligence (XAI) has emerged as a direct response to the "black box" problem, encompassing a range of techniques aimed at making AI system decisions interpretable and understandable to all relevant stakeholders, from security analysts to end-users (Barredo Arrieta et al., 2020). Establishing a structured taxonomy is essential given the diversity of XAI methodologies. Such taxonomies typically classify approaches along several

foundational dimensions: intrinsic vs. post-hoc methods, model-specific vs. model-agnostic techniques, and global vs. local explanatory scopes. Because modern phishing classifiers involve highly complex architectures, the literature has predominantly emphasised post-hoc, model-agnostic methods. These techniques can generate explanations for any pre-trained model by providing local, instance-level interpretability. Consequently, the central question they address becomes: “Why has this email been flagged?”

There are two methods in this category that have become integral to basic theory due to their strong versatility. LIME (Local Interpretable Model-Agnostic Explanations) operates on a straightforward and intuitive principle: it generates numerous small perturbations of the instance of interest and, by learning a simple, interpretable model from these perturbations, explains the prediction of interest from this black box model (Ribeiro, Singh, & Guestrin, 2016). In contrast, SHAP (Shapley Additive exPlanations) is more theoretically sound as it's built on cooperative game theory. It accurately measures each feature's contribution to a prediction by thinking of them as players in a game. Thus, it fairly and accurately distributes their contribution to the final outcome (Lundberg & Lee, 2017).

Phishing detection models have successfully employed these techniques to demystify decisions in the literature and, on occasion, boost performance as well. Research often combines top-performing ML models with these XAI frameworks to achieve both accuracy and interpretability simultaneously. For instance, Akhtar et al. (2025) and Fatima et al. (2025) provide conclusive evidence with the results show that with the use of LIME and SHAP, it can be represented as an explanation of individual predictions with a very high accuracy level, around 98-99% by models such as XGBoost and Random Forest (Akhtar et al., 2025, Fatima et al., 2025). More recent works by Hernandes Jr. et al. (2021) considered a post-hoc explanation

using LIME with a Random Forest and the use of intrinsic, interpretable models, such as the Explainable Boosting Machine (EBM) (Galego Hernandes et al., 2021). Shafin (2025) adopted a rather innovative methodology, essentially developing a framework (SLA-FS) that utilizes SHAP and LIME not just for interpretation, but also for intelligent feature selection that was capable of increasing the accuracy of a Random Forest model by 0.65% (Shafin, 2025). The results of these and other major studies are summarized in Table 4, where the pairing of high-performance ML models with well-established XAI techniques is clearly observable.

Table 4. Summary of Representative Studies Integrating ML and XAI for Phishing Detection

Author Names	Methods	Performance Metrics
Galego Hernandes et al., 2021	1. Random Forest + LIME 2. Explainable Boosting Machine (EBM)	Accuracy: RF: %97.32 EBM: %96.46
Akhtar et al., 2025	XGBoost, LightGBM, RF, KNN, CNN + LIME	XGBoost Accuracy: 99.65%
Shafin, 2025	RF, XGBoost + SHAP & LIME	RF Accuracy: 97.41%
Fatima et al., 2025	Random Forest (RF) + LIME & SHAP	RF Accuracy: 98.35%

Even though these studies prove the capability of XAI, a critical review of this body of work reveals significant research gaps. Most existing literature has focused on demonstrating the effectiveness of XAI, but important operational questions remain unanswered. Aspects including the computational burden of producing explanations in real-time, their empirical benefits, their practical utility to a security analyst, and their inherent robustness to adversarial manipulation are still relatively uninvestigated and unevaluated (AbdulNabi & Yaseen, 2021). As a result, the core question on the ability of phishing models to explain themselves will be answered. Furthermore, many people will work to make these

explanations usable, useful and secure for practical use. Thus, this will be the current trend, challenge, and future of this field.

Trends in Explainable AI for Phishing Detection

The basic techniques of LIME and SHAP showed that the black box of phishing detection models can be opened. However, the field of Explainable Artificial Intelligence is evolving rapidly beyond these early proofs of concept and is now facing a crucial question: "How do we make the explanations transparent, but also useful and integrated into a security analyst's workflow?" This section will examine the ongoing trends that will shape the next-generation explainable security systems.

These trends have evolved from a mere description of simple properties to useful, efficient, usable, and actionable knowledge. Importantly, this change came about because continuous, real-time explanations can aid security analysts dealing with changing adversarial campaigns but static descriptive ones often fail. Given this, four main vectors of innovation are in focus: (i) counterfactual and actionable explanatory mechanisms, (ii) interactive and human-centred interfaces, (iii) renewed interest in inherently interpretable architectures, and (iv) robust engineering solutions that target the technical feasibility of XAI in real-world applications.

Beyond Feature Attributions: Towards Actionable Explanations

The first major trend highlights a conceptual shift in what counts as a good explanation. Attribution-based methods, such as SHAP and LIME, which are traditional, are effective in answering the question: which features were most important for this prediction? We get a ranked list or a visual graph. For example, the presence of a suspicious sender field and the word invoice were the two most important features that made the algorithm flag the email. Although

this information is useful for auditing a model's decision, it may not be so useful to a pressured analyst.

The new view of actionable explanations asks an easier question: “What is the minimum change to the input that would change the model’s decision?” Formally, this is known as generating counterfactual explanations (Wachter, Mittelstadt, & Russell, 2018). Telling what wouldn’t have happened if the individual hadn’t received the email is far more powerful than explaining why it is phishing. For a phishing email, it could be posited.

"This email would be classified as legitimate if the sender's domain were from a known business partner (e.g., @partner.com instead of @pàrtner.com) and the anchor text matched the visible URL.".

Human-centric nature of counterfactuals is probably what makes them effective. We can identify the important factors that alter the scale, resulting in a pattern that aligns with human natural reasoning about cause and effect. As a security analyst, these insights are considered highly actionable. A list of suspicious features is not all that is provided; in addition to this, by identifying the ‘tipping point’ of the model’s logic, the confirmation of whether an artefact is indeed malicious is made easier or the identification of the specific feature causing a false positive in a legitimate email is made easier. As a result of this trend, XAI shifts from being a passive reporting tool to a proactive diagnostic assistant in cybersecurity workflows.

Human-Centric and Interactive Explainability

Although methods like LIME and SHAP have dominated the XAI field, a strong countertrend with increasing momentum has emerged in the form of the development and adoption of inherently interpretable or “transparent box” models. In general, it is preferable to use a transparent model by design, if it can be made to display competitive performance, rather than using computational resources

to approximate the behaviour of a black-box model (Nori, Jenkins, Koch, & Caruana, 2019). This trend challenges the standard belief that there is an inevitable trade-off between accuracy and interpretability.

The Explainable Boosting Machine (EBM) is perhaps the best-known example of this trend in modern machine learning. An Explainable Boosting Machine (EBM) is a Generalized Additive Model (GAM), and its performance is very competitive with other popular models like Random Forest and XGBoost because it uses gradient boosting. Unlike these models, however, EBMs are completely transparent in their own unique way.

This is achieved by:

- Decomposition: A model is created for every feature to capture its contribution towards the final prediction. The development of multiple testing facilities, like those for Talc, PCB/PAH, and cotton cultivation, has strengthened the testing capacities in the country. The resulting prediction is the sum of the calculated values of feature functions. This structure contains no complex, hidden relations among features, as it makes them clear.
- Direct Visualisation: The contribution of each individual feature can be visualised using 2D plots. For example, we may analyse the plot corresponding to the ‘URL Length’ feature to understand how changes in input length affect the risk score generated by the system. This visual clarity is particularly useful for auditing model behaviour for different threshold faults and facilitates the quick identification of non-linear risk patterns that aggregate metrics may conceal. In this case, the graph doesn’t spoil the behaviour! It is the model for that characteristic. As a result, the visualization is made fully faithful, free from

the estimation errors or information loss characteristic of surrogate explanation methods.

This approach offers a significant benefit for phishing detection. Therefore, an analyst does not require a separate post-hoc tool to understand a decision. The internal graphs of an EBM can be examined and interpreted directly to assess the contribution to risk of the individual components of the email (i.e. whether the email contains a certain keyword, the domain name of the sender, the number of exclamation marks, etc.) By default, the model's reasoning is completely transparent and auditable.

Phishing has often posed a problem for interpretable models. In the past, it has proven too complex for interpretable models to capture the essence of the issue. However, modern transparent box models offer promise. EBMs are one such model. Furthermore, EBMs provide striking evidence that full transparency does not have to come at the cost of high accuracy. The latest trend represents a fundamental paradigm shift. This refers to the shift from explaining black boxes to the *de novo* design of high-performance, transparent models.

Performance-Aware and Real-Time XAI

The recent trend aims to address a significant practical issue that hinders the widespread adoption of XAI in deployed cybersecurity, specifically in terms of computational performance. Generating a detailed SHAP explanation for a single prediction may be possible in an offline research environment. However, creating SHAP explanations for thousands of emails per second in an active detection pipeline is a challenging engineering problem. Accordingly, there is a clear shift towards performance-sensitive XAI. Moreover, more and more explanation methods are designed and optimised to work under the latency constraints of production systems (Štrumbelj & Kononenko, 2014).

A basic challenge faced is that many theoretically sound XAI methods (those based on extensive input perturbation, such as KernelSHAP) can be expensive. Obtaining even a single explanation using existing approaches requires querying the underlying black-box model hundreds or thousands of times, which introduces unacceptable overheads. This current trend aims to bridge the gap between theory and practice by introducing specific methods.

- Fast Approximation Algorithms: There is increasing research effort driving advanced algorithms that can approximate top-performing algorithms like SHAP more quickly. Because calculating the exact Shapley value involves high latency, which makes it unsuitable for real-time network filtering, our earlier work on fast approximations is now an important operational need. Using smart sampling or structures within a model can lessen the number of simulations, thus generating a quality explanation in comparatively fewer resources. As a result, instead of stochastic perturbation of features, these methods offer rapid isolation of influential features. This ensures stability in explanations without compromising on speed.
- Hardware Acceleration: Similar to how GPU inclusion and specialised hardware have fast-tracked the inference of deep learning tasks, performance-oriented XAI architectures are also increasingly parallelised to speed up explanation generation. This compatibility exists because perturbation-based methods require the simultaneous evaluation of independent input variants, which is well-suited to the vector processing units of current accelerators. As a result, prediction and perturbation processes are treated as highly parallelizable workloads. Therefore, spreading them out across multi-

core systems or GPU threads significantly reduces the latency overhead of real-time interpretability compared to a single GPU.

- Asynchronous Explanation Generation: In operational security workflows, immediate (blocking or allowing) classifications must happen in microsecond time frames. On the other hand, the detailed analytical explanation may have a delay of seconds to minutes after the alert. In light of this temporal mismatch of a few seconds, several architectural frameworks have been proposed that decouple the generation of explanations from the primary predictor. According to this approach, the inference takes place in real-time. Then, an asynchronous request is sent to the explanation service on the occurrence of a suspicious event to compute an explanation as a background process. This stratification of architecture is crucial so that the feature attribution logic, which is expensive, does not slow down the throughput of the primary firewall or gateway. Thus, the main detection pipeline operates unaffected while comprehensive explanations are available later for forensic purposes.

It's essential to modify operations; otherwise, XAI frameworks would be limited to offline analysis and post-hoc debugging. They should not become mere accessories, but rather ought to be enshrined within real-world security operations. Due to the short lifecycle of present-day phishing infrastructure, which is only accessible for a couple of hours, forensic interpretation offers very little help to defenders in taking timely actions. Performance-aware architectures bring transparency and contemporary cybersecurity together in harmony. As a result, the necessary operational speed and scalability are maintained. The system uses

codes optimized through algorithmic modelling and intelligent design for achieving this equilibrium.

Current Challenges in Explainable AI for Phishing Detection

Although the technologies outlined above appear highly promising, operationalizing Explainable Artificial Intelligence (XAI) in real-time threat detection environments remains a significant challenge. These difficulties extend beyond implementation logistics and encompass substantial technical and methodological obstacles, forming an active area of academic inquiry. A central source of friction is the semantic gap—the disconnect between the mathematically derived explanations produced by XAI methods and the higher-level threat intelligence that analysts require to make rapid, contextually informed decisions. Mitigating these constraints is essential to ensuring the robustness of XAI-enabled frameworks. Unless these structural issues are resolved, such systems cannot be effectively integrated as trusted decision-support resources within Security Operations Centres (SOCs).

The Accuracy-Interpretability Trade-off Revisited

Within the machine learning domain, a longstanding and widely acknowledged challenge is the trade-off between accuracy and interpretability. Models that achieve the highest predictive performance are typically the most opaque, and the reverse is often also true. Although Explainable Boosting Machines (EBMs) offer a notable exception by providing strong accuracy while maintaining interpretability, the trade-off remains a persistent and practical constraint in state-of-the-art phishing detection systems. Nowadays, phishing classifiers, which are highly efficient, are built on large, complex deep learning architectures, such as Transformers. This model excels at detecting subtle details in sophisticated phishing attacks that are difficult to identify. They are better than simpler

models by a small margin. In security, this difference matters significantly because a 0.1% detection improvement can prevent thousands more threats – even for a medium-sized business. The security teams now face a dilemma. They must decide whether to opt for the slightly less accurate but comprehensible "glass-box" model or select the more accurate "black-box" model and rely on later XAI methods. Generally, maximum protection is preferred when decisions are made in a high-stakes operational environment. This necessitates further requirement of more tools and more reliable ones of them.

Ensuring the Quality and Reliability of Explanations

One of the biggest obstacles in XAI is that generating an explanation does not make it good or reliable. A seemingly complete explanation might be misleading and lead an analyst to erroneous diagnosis of a threat. Various elements represent this challenge.

The Fidelity Problem: Post-hoc explanation methods design a separate and simpler model after the fact to approximate the behaviour of the complex black-box model. This is the fidelity problem. Fidelity refers to how faithfully the approximation mirrors the internal logic of its model. Within this context, we face a trade-off. For an explanation to be simple enough for humans to understand, it often must abstract away or simplify details from the model we started with. This can lead to a loss of accuracy.

Robustness and Stability: The explanation should be robust. This means that if the explanation undergoes a minor, semantically irrelevant change, then it should not change drastically and should remain stable. However, research has shown that many commonly used explanation methods are easily vulnerable and sensitive to small perturbations (Alvarez-Melis & Jaakkola, 2018). The instability, by its nature, may seriously undermine an analyst's

confidence, as it indicates that explanations may be attached to superficial artefacts rather than real signals.

Adversarial Attacks on Explanations: This challenge is especially critical in security, where attackers may manipulate inputs to corrupt or mislead the model's explanations. Skilled attackers can write phishing emails that avoid detection. They can also infiltrate the explanation process and compromise it. According to several investigations, it has been established that misleading explanations can be obtained using LIME and SHAP to generate specially designed inputs (Slack, Hilgard, Jia, Singh, & Lakkaraju, 2020. This new threat makes something useful into a weapon that can deceive us.

These technical complications are further complicated by the difficulty of evaluation. While we can easily measure the predictive power of a model, measuring the 'quality' of an explanation is much more difficult. Currently, there is no agreed-upon metric, similar to F1-score, for explainability. The best way to evaluate an explanation is through costly human-centred studies. However, this "gold standard" evaluation has many practical problems.

- Conducting studies that need domain experts is very costly and takes a lot of time and effort. So, they are quite expensive.
- When doing these evaluations, there are difficulties in recruiting and engaging a large number of security analysts. This inhibits the broader generalization of findings.
- What is a good explanation can be quite subjective and can change greatly from one evaluator to another, and from one context to another.

Often, when system evaluation is attempted, anecdotal evidence is utilized. However, this is not necessarily beneficial, as the advancement toward a proper science of XAI is impeded. Ultimately, the objective is to possess effective and reliable XAI systems; however, their performance cannot be ascertained without proper evaluation.

The Intractable Problem of Evaluation

There are still a number of practical issues that remain (even if all the technical and methodological issues are resolved completely). These practical issues are, namely, scalability and integration.

In real life, emails are not often very simple (or complex) as they tend to have different layouts, HTML content, images, and attached files. Researchers are therefore currently working on XAI frameworks that can generate credible and reliable explanations for multimodal data. A major challenge in this area remains aligning feature attribution, wherein visual anomalies within an image must be contextually linked to associated words to form a coherent narrative of threat. Furthermore, XAI tools cannot only be used in isolation but also need to be integrated efficiently in the SOC workflow and the overall SIEM and SOAR ecosystem.

This necessity introduces several engineering challenges, including the standardization of APIs for data exchange, the seamless integration of UI/UX components into existing analyst dashboards, and the maintenance of high-throughput data pipelines. The effectiveness of these systems depends not only on algorithmic performance but also on governance and engineering discipline, ensuring that they remain scalable, interoperable, and fully integrated into the daily workflows of security professionals.

Future Directions

Addressing the multifaceted nature of explainability in phishing detection demands a future research agenda that extends beyond incremental methodological improvements. The trajectory of XAI in cybersecurity encompasses more than algorithmic efficiency; it aims to establish a holistic human–machine partnership. It also builds a holistic partnership. Due to the fact that phishing attacks primarily target human cognitive weaknesses, future defend will not act as a stand-alone mechanism, but as a partner augmenting human analyst decision-making. In light of this, this section presents important future research directions. There is a need to strengthen the system's security while also enriching the context of explanations. The progress of XAI practices depends on the advancement of human-centred design frameworks and evaluation metrics, as well as related works.

Technical Frontiers: Towards Secure, Multimodal, and Context-Aware Explanations

A significant frontier in XAI research is the technical robustness of the explanations themselves. Future research should aim to create an XAI whose explanations cannot be altered by adversarial input. It is essential to regard explanatory robustness as a first-rate security property, where state-of-the-art systems are alarmingly fragile and can be manipulated by adversaries (Slack, Hilgard, Jia, Singh, & Lakkaraju, 2020, Dombrowski, Anders, Müller, & Kessel, 2022). A cunning adversary could create a malicious artefact that can lead to a plausible and incorrect feature attribution, where a human can override a correct decision, thinking it is incorrect. Future methods will need to be designed to be inherently stable and insensitive to unwanted disturbances. It will continuously monitor the explanations for concept drift and develop models specialized for the downstream detection of deceptive

rationales. By recognising that the explainability layer is also an attack surface, we can make XAI systems ready to be truly explainable and robust.

Aside from security features, the future of XAI focuses on making explanations more intelligent and comprehensive. Phishing attacks are becoming increasingly sophisticated, utilising multiple modes and cleverly employing images, attachments, and complex narratives.

There is a future need to develop techniques that can generate a coherent explanation for a choice resulting from different sources of data, rather than relying solely on text. This refers to a description of the feature fusion processes, such as the forged logo and the overlaying text message. Next-gen XAI systems will use your personal info to provide personalized and contextual explanations, like a flag on a suspicious email stating it's 'highly unusual for you' based on your unique experience. We envisage that XAI will help us create a basic model explainer that transforms the threat intelligence analyst into a complete threat narrative.

Making XAI Work: Putting Humans in the Loop and Evaluating it Carefully

One of the important future directions is the growing operational practice of XAI, which will move XAI from the lab to a reliable and measurable practice in the SOC. To this end, it is suggested to view the process as a human-centred dialogue leading to co-adaptive systems and mutual learning for humans and AI agents (Miller, 2019). Interfaces with strong feedback loops will facilitate analyst validation or correction of an explanation, allowing the analyst to become a proactive instructor. In co-adaptive learning loops, the feedback received is leveraged as a training signal to update both the detection model and the explanation. This loop thus expresses the true power of this approach. This view frames

explainability as the sole mechanism by which humans and machines develop their emergent partnership.

It is suggested that for XAI to evolve into a proper engineering discipline, it must address the hard evaluation problem by establishing a proposed ground zero (Doshi-Velez & Kim, 2017). In reply, landmark organisations, like NIST, want more methodological rigour. This supposes several things. It will be very important to create datasets that incorporate expert-annotated ground-truth explanations. When benchmarks do not exist, estimates of explanatory fidelity will often rely on qualitative user studies that are hard to standardise and reproduce in different settings. The wider scientific community should take action to encourage reproducible science by running shared tasks with standardised protocols which allow for fair comparisons between different XAI methods. If we want progress to last, we must define how we will measure performance, trust, and cognitive load in a human-centric manner. As this scaffolding is established, the data field will shift toward quantitative assessment, culminating in a future time when an explanation's quality will be as verifiable as a prediction's accuracy.

Conclusion

The paragraph provided a detailed account of the evolving nature of XAI in relation to phishing detection. Machine learning classifiers played a crucial role in the fight against phishing email attacks. In fact, white-box models provide a level of transparency to end users. However, the constant adaptability of phishing attacks has led to a transition of phishing classifiers. As a result of this transition, attackers now employ a spectrum of threats. These threats range from transparent yet rule-based systems to opaque classifiers of high efficacy.

Despite the improvement in detection accuracy, this advancement concurrently introduced the "black box" phenomenon,

which diminished model interpretability and constrained the effectiveness of security analysts' decision-making processes. When the reasoning behind a model's alert cannot be examined, such alerts remain unverified data points rather than actionable intelligence, thereby undermining the core objective of developing an advanced detection architecture. In this context, the relevance of Explainable AI (XAI) has become increasingly apparent, particularly through techniques such as LIME and SHAP, which are employed to enhance transparency and interpretability. Nevertheless, the field continues to face significant challenges, including the adversarial robustness of explanations—a complex issue in its own right—and the lack of standardized evaluation protocols.

The findings of this review indicate that explainability in phishing detection has evolved from a secondary performance concern into a central component of modern cyber-resilience strategies. What was once viewed primarily as an academic challenge is now firmly embedded in the operational reality of organizations. Explainable AI functions as the connective layer between the technical sophistication of contemporary machine learning models and the practical decision-making processes of security analysts. Without this layer, alert fatigue becomes unavoidable, and analysts remain confined to reactive triage rather than advancing toward informed threat validation and more deliberate response actions. Ultimately, the future of phishing defence does not depend solely on algorithmic accuracy; it hinges on an effective human–machine partnership in which trust, interpretability, and transparency form the basis of an adaptive and sustainable security posture.

Usability will be an important quality attribute in future security architectures, and it will be non-negotiable. In the future, defence mechanisms will be judged not only on predictability but also on whether they allow people to understand how they work.

Thus, concentrating on explainability in phishing detection provides a launchpad for the broader adoption of AI in the cybersecurity ecosystem. The detection of phishing shows that an embedded autonomous system is competent and reliable. Read on to understand phishing detection. We must strengthen our technical, trustworthy, transparent, and collaborative frameworks to protect the digital space, which is becoming increasingly important – both economically and geopolitically – in the 21st century.

References

AbdulNabi, I., & Yaseen, Q. (2021). Spam email detection using deep learning techniques. *Procedia Computer Science, 184*, 853–858. https://doi.org/10.1016/j.procs.2021.03.107

Akhtar, H. M. U., Nauman, M., Akhtar, N., Hameed, M., Hameed, S., & Tareen, M. Z. (2025). Mitigating cyber threats: Machine learning and explainable AI for phishing detection. *VFAST Transactions on Software Engineering, 13*(2), 170–195. https://doi.org/10.21015/vtse.v13i2.2129

Alvarez-Melis, D., & Jaakkola, T. S. (2018). *On the robustness of interpretability methods.* http://arxiv.org/abs/1806.08049

Alsuwit, M. H., Haq, M. A., & Aleisa, M. A. (2024). Advancing email spam classification using machine learning and deep learning techniques. *Engineering, Technology & Applied Science Research, 14*(4), 14994–15001. https://doi.org/10.48084/etasr.7631

APWG, "APWG Phishing Activity Trends Report," [On-line]. Available:https://apwg.org/trendsreports/. [Accessed: Apr. 17, 2025]

Asliyuksek, H., Tonkal, O., & Kocaoglu, R. (2025). A comparative evaluation of a multimodal approach for spam email classification using DistilBERT and structural features. *Electronics, 14*(19), 3855. https://doi.org/10.3390/electronics14193855

Barredo Arrieta, A., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion, 58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Chandan, J. R., Dsouza, G. E., George, M., & Bhadra, J. (2023). Spam message filtering based on machine learning algorithms and BERT. In *Proceedings* (pp. 227–238). https://doi.org/10.1007/978-981-19-1844-5_19

Dedeturk, B. K., & Akay, B. (2020). Spam filtering using a logistic regression model trained by an artificial bee colony algorithm. *Applied Soft Computing, 91*, 106229. https://doi.org/10.1016/j.asoc.2020.106229

Dombrowski, A.-K., Anders, C. J., Müller, K.-R., & Kessel, P. (2022). Towards robust explanations for deep neural networks. *Pattern Recognition, 121*, 108194. https://doi.org/10.1016/j.patcog.2021.108194

Doshi-Velez, F., & Kim, B. (2017). *Towards a rigorous science of interpretable machine learning.* http://arxiv.org/abs/1702.08608

Fatima, Z., et al. (2025). Explainable AI for IoT devices and robotic communication phishing detection. *Engineering, Technology & Applied Science Research, 15*(5), 26478–26486. https://doi.org/10.48084/etasr.11595

Galego Hernandes, P. R., Floret, C. P., Cardozo De Almeida, K. F., Da Silva, V. C., Papa, J. P., & Pontara Da Costa, K. A. (2021). Phishing detection using URL-based XAI techniques. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1–6). IEEE. https://doi.org/10.1109/SSCI50451.2021.9659981

Hohman, F., Head, A., Caruana, R., DeLine, R., & Drucker, S. M. (2019). Gamut. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). ACM. https://doi.org/10.1145/3290605.3300809

Junnarkar, A., Adhikari, S., Fagania, J., Chimurkar, P., & Karia, D. (2021). E-mail spam classification via machine learning and natural language processing. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)* (pp. 693–699). IEEE. https://doi.org/10.1109/ICICV50876.2021.9388530

Khaleel, A., & Salihi, A. (2019). *Spam detection by using word-vector learning algorithm in online social networks.*

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems, 30*, 4766–4775. https://doi.org/10.48550/arXiv.1705.07874

Meléndez, R., Ptaszynski, M., & Masui, F. (2024). Comparative investigation of traditional machine-learning models and transformer

models for phishing email detection. *Electronics, 13*(24), 4877. https://doi.org/10.3390/electronics13244877

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267*, 1–38. https://doi.org/10.1016/j.artint.2018.07.007

Nasreen, G., Murad Khan, M., Younus, M., Zafar, B., & Kashif Hanif, M. (2024). Email spam detection by deep learning models using novel feature selection technique and BERT. *Egyptian Informatics Journal, 26*, 100473. https://doi.org/10.1016/j.eij.2024.100473

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). *InterpretML: A unified framework for machine learning interpretability.* https://doi.org/10.48550

Rayan, A. (2022). Analysis of e-mail spam detection using a novel machine learning-based hybrid bagging technique. *Computational Intelligence and Neuroscience, 2022*, 1–12. https://doi.org/10.1155/2022/2500772

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). ACM. https://doi.org/10.1145/2939672.2939778

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications, 117*, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). *Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods.* http://arxiv.org/abs/1911.02508

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems, 41*(3), 647–665. https://doi.org/10.1007/s10115-013-0679-x

Tida, V. S., & Hsu, S. (2022). *Universal spam detection using transfer learning of BERT model.* http://arxiv.org/abs/2202.03480

Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. https://doi.org/10.48550/arXiv.1711.00399

Wang, Y., Zhu, W., Xu, H., Qin, Z., Ren, K., & Ma, W. (2023). A large-scale pretrained deep model for phishing URL detection. In *ICASSP 2023 – IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICASSP49357.2023.10095719

Whittaker, C., Ryner, B., & Nazif, M. (2010, February). Large-Scale Automatic Classification of Phishing Pages. In Ndss (Vol. 10, p. 2010).

Yakandawala, Y. L. D. H., Madushanka, M. K. P., & Ilmini, W. M. K. S. (2024). Explainable AI for transparent phishing email detection. In *2024 International Conference on Advances in Technology and Computing (ICATC)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICATC64549.2024.11025302

BÖLÜM 2

FACTORS AFFECTING THE PERFORMANCE OF ARTIFICIAL INTELLIGENCE MODELS: A THEORETICAL REVIEW

İsmail AKGÜL[1]

1. Introduction

Machine learning and, in particular, deep learning models constitute the backbone of today's AI-based cognitive systems. Over the last decade, human-level and in some cases even super-human performance has been reported in many domains, including medicine, defense, education, transportation, and industry (LeCun et al., 2015).

Recent advances, especially around large-scale deep networks and foundation models, have made the scaling relationships between data, model, and computational resources more visible. When model size, data quantity, and compute budget are scaled jointly, performance has been shown to increase in a predictable manner, which implies that the performance of AI

[1] Assoc. Prof. Dr., Erzincan Binali Yıldırım University, Faculty of Engineering and Architecture, Department of Computer Engineering, Orcid: 0000-0003-2689-8675

systems should be approached as a multidimensional resource allocation problem (Kaplan et al., 2020).

In this context, in addition to the traditional “model-centric” paradigm that focuses on architectural design, a “data-centric” AI perspective has come to the fore, emphasizing systematic improvement of the data across its entire lifecycle. Data-centric AI studies show that substantial performance gains can be obtained by improving data quality, diversity, and data maintenance processes, underlining that data engineering is at least as critical as model architecture (Malerba and Pasquadibisceglie, 2024; Zha et al., 2025).

Similarly, it is reported that when practical constraints such as data scarcity, imbalanced data distributions, and noisy labels are mitigated through techniques such as data augmentation, active learning, and other data-centric approaches, notable improvements can be achieved in the performance of deep learning systems (Alzubaidi et al., 2023; Bhatt et al., 2024).

Overall, the performance of AI models is shaped by the interaction of many components, including data quality and diversity, preprocessing and data augmentation procedures, model architecture, training strategies and hyperparameter settings, regularization approaches, evaluation metrics, and computational resources. The aim of this chapter is to provide a holistic discussion of the key factors that enhance or limit the performance of AI models, without going into implementation details.

2. Data-Related Factors

The performance of an AI model depends to a large extent on the properties of the data used for training and testing, particularly on data quality, data quantity, data diversity, and class imbalance.

2.1. Data Quality

Data quality is one of the most critical components determining the correctness and reliability of the information that a model can learn. Erroneous, incomplete, noisy, or inconsistent data prevent the model from capturing the underlying patterns accurately and directly lead to performance degradation. Common data quality problems in practice include:

- Noisy or incorrect labels
- Missing values
- Class imbalance
- Low-resolution or corrupted images
- Sampling bias

It has been shown that even in very large and widely used datasets such as ImageNet, the proportion of mislabeled samples can be in the range of 6–10%, which substantially reduces the predictive power of models and leads to erroneous generalizations and lower-than-expected performance, especially for fine-grained classes (Northcutt et al., 2021).

2.2. Data Quantity

AI models, particularly deep learning architectures, often contain millions of parameters and therefore require large amounts of data. The relationship between data quantity and model performance is commonly formulated within a power-law framework (Hestness et al., 2017). In general, this relationship can be summarized as follows:

- As the amount of data increases logarithmically, accuracy improves in a non-linear but steady manner.

- When data are insufficient, a significant proportion of model parameters cannot be effectively learned, and the risk of overfitting increases.

Thus, logarithmic increases in data size lead to continuous and meaningful improvements in model performance. In deep architectures such as ResNet, EfficientNet, and Transformer-based models, increasing the amount of data is a key factor for both training stability and generalization performance in real-world scenarios.

2.3. Data Diversity

Data diversity, that is, the degree to which different scenarios, contexts, and sample types are covered, is a key determinant of the model's capacity to generalize. Datasets that include different viewpoints, lighting conditions, environments, and individuals enable the model to adapt not only to the training distribution but also to the broader variability encountered in the real world. For example:

- Face recognition systems require face images from different age groups, ethnic backgrounds, poses, and lighting conditions.
- Object recognition systems exhibit better generalization when trained on images of objects captured at different scales, backgrounds, and camera angles.

When data diversity is low, the model tends to overfit the narrow set of examples seen during training and fails in new, previously unseen situations. This not only increases the risk of overfitting but also gives rise to systemic biases if the dataset has limited representational power (Shorten and Khoshgoftaar, 2019).

2.4. Class Imbalance

Class imbalance refers to the situation where the number of samples belonging to different classes in a dataset differs significantly. In other words, any dataset that exhibits a non-uniform distribution across classes can technically be considered imbalanced. Such imbalances cause the model to focus predominantly on majority classes during training and to "ignore" minority classes. As a result:

- Accuracy and especially recall for rare classes decrease.
- The number of false negatives increases.
- The reliability of the model in real-world scenarios is weakened.

To mitigate these adverse effects, various strategies have been proposed in the literature at both the data level and the algorithmic level. At the data level, undersampling of the majority class, oversampling of the minority class, and synthetic sample generation methods such as SMOTE are widely used (Chawla et al., 2002). At the algorithmic level, class weighting, cost-sensitive learning, and decision threshold adjustment are among the most prominent approaches. More broadly, strategies for dealing with class imbalance in AI are discussed within a wide framework that includes data-level resampling, algorithm-level adjustments and ensemble methods (He and Garcia, 2009).

3. Data Preprocessing and Data Augmentation

The performance of an AI model depends not only on the quality of the raw data but also on how the data are processed before being fed into the model. In this respect, preprocessing steps such as scaling, normalization, and standardization, as well as data augmentation strategies, are fundamental components that shape

both the stability of training and the generalization performance of the model.

3.1. Normalization and Standardization

For AI models, particularly deep networks trained with gradient-based methods, faster and more stable convergence largely depends on input variables having similar scales. Features with different orders of magnitude can complicate the optimization process, slow down learning, and cause the effects of some parameters to dominate others. Therefore, various normalization and standardization techniques are employed to reduce scale differences among features. Common approaches can be summarized as follows:

- Min–max scaling: rescales each feature to a predefined range (usually [0, 1] or [−1, 1]).
- Z-score standardization: transforms features so that they have approximately zero mean and unit variance.

In addition to these input scaling methods, Batch Normalization, which normalizes intermediate activations at the mini-batch level in deep neural networks, is also widely used. Batch Normalization reduces internal covariate shift, shortens training time considerably, and enables the safe use of higher learning rates (Ioffe and Szegedy, 2015). Consequently, appropriate normalization and standardization strategies are essential preprocessing steps that directly affect both the stability of optimization and overall model accuracy.

3.2. Data Augmentation

Data augmentation is a technique that aims to increase the effective amount of data by generating new and diverse samples derived from the existing dataset. In situations where data collection is costly, time-consuming, or ethically constrained, data

augmentation plays a critical role in reducing overfitting and improving the generalization capacity of the model. Basic image data augmentation techniques include:

- Rotation and flipping
- Noise injection
- Brightness and contrast adjustments
- Random cropping

These conventional transformations modify the geometric and photometric properties of images in a controlled manner, making the model more robust to different variations. Classical data augmentation strategies have been shown to consistently improve performance in many tasks, such as image classification and object recognition, especially for small and medium-sized datasets (Shorten and Khoshgoftaar, 2019).

In addition to traditional methods, more advanced augmentation approaches have gained significant attention in recent years. GAN-based augmentation and diffusion models enrich the data space by generating synthetic samples that resemble the real data distribution, while techniques such as Mixup (Zhang et al., 2017) and CutMix (Yun et al., 2019) perform mixing operations on samples and labels, smoothing decision boundaries, and providing a better-behaved hypothesis space. These modern methods offer additional benefits beyond classical augmentations, particularly in deep convolutional and Transformer-based architectures.

4. Model Architecture-Related Factors

The performance of an AI model depends not only on data-related properties but also on the chosen model architecture and its design details. Model depth, number of parameters, types of layers, activation functions, and regularization strategies are key

architectural factors that determine both the representational power and the generalization ability of the model.

4.1. Model Depth and Number of Parameters

Model depth (number of layers) and the total number of parameters are primary architectural components that determine the complexity of the representations that can be learned. In general, deeper and more parameter-rich networks are capable of learning more complex and abstract patterns in the data. However, increasing depth and parameter count also bring several challenges:

- Computational cost (training time and memory usage) increases.
- The risk of overfitting grows; the model may fit the training data too closely and perform poorly on new data.
- In very deep networks, optimization issues such as vanishing and exploding gradients may arise.

To address these problems, architectures with residual connections, such as ResNet were developed, which significantly improve the trainability of very deep networks (He et al., 2016). Residual blocks establish skip connections between layers, facilitating gradient flow and enabling stable training of networks with hundreds of layers.

4.2. Layer Types

Different problem types benefit from different layer structures. Therefore, when designing a model, the types of layers should be chosen in accordance with the nature of the task. Common layer types and their typical application areas can be summarized as follows:

- Convolutional (CNN) layers: used primarily in vision and 2D signal processing tasks to capture local spatial patterns such as edges, textures, and shapes.
- Recurrent / LSTM / GRU layers: preferred for sequential data such as time series, text, and speech, enabling the modelling of temporal dependencies and long-term relationships.
- Transformer encoder layers: widely used in natural language processing, time series analysis, and increasingly in vision, thanks to self-attention mechanisms that effectively model long-range dependencies among input tokens.
- Fully connected (dense) layers: typically placed at the end of the network to map learned representations to final outputs such as class probabilities or regression targets.

As an example of this diversity, Vision Transformer (ViT) offers an alternative to classical CNN-based approaches for image classification and achieves competitive or even superior performance at sufficient data scales (Dosovitskiy et al., 2021). Thus, the appropriate combination of layer types directly affects both the expressive power of the model and its fit to the target problem.

4.3. Activation Functions

Activation functions provide non-linear transformations between layers, enabling neural networks to learn complex, high-order functions. The choice of activation function can directly influence training dynamics (gradient flow, convergence speed) and final performance. Some widely used activation functions include:

- Rectified Linear Unit (ReLU): due to its simplicity and computational efficiency, it is the most commonly used

activation function in deep networks; it passes positive values unchanged and sets negative values to zero.

- Leaky ReLU: a generalization of ReLU with a small non-zero slope in the negative region, aimed at alleviating the "dead neuron" problem.
- Gaussian Error Linear Unit (GELU): an activation function that scales inputs in a probabilistic manner and has become popular, particularly in Transformer-based models (Hendrycks, 2016).
- Mish: a continuously differentiable and smooth activation function that has been reported to yield better performance than ReLU variants in some architectures (Misra, 2019).

In summary, choosing an appropriate activation function is a crucial architectural decision that supports stable gradient flow, accelerates learning, and helps the model learn richer representations.

4.4. Regularization Methods

Regularization refers to methods that aim to prevent the model from overfitting the training data and thus to increase generalization performance. In high-capacity deep models, the risk of overfitting is substantial, so it is common practice to employ multiple regularization techniques in a balanced manner. Major regularization methods can be summarized as follows:

- Dropout: temporarily disables randomly selected neurons during training to prevent the network from relying too heavily on specific activations and to encourage the learning of more robust representations (Srivastava et al., 2014).

- Weight decay (L2 regularization): penalizes large weight magnitudes and thus promotes smoother decision boundaries.
- Early stopping: monitors validation loss or accuracy and stops training once performance deteriorates, providing a simple yet effective way to reduce overfitting.
- Data augmentation: although implemented at the data level, it acts as a strong form of regularization by exposing the model to diverse input variations.
- Batch normalization: normalizes layer activations and, in addition to accelerating training, introduces a mild regularization effect.
- Label smoothing: replaces perfectly sharp one-hot targets with slightly smoothed distributions, reducing the tendency of models to produce overconfident predictions and often improving generalization (Szegedy et al., 2016).

The appropriate combination of these techniques helps preserve model capacity while preventing convergence to overly complex hypotheses, leading to balanced performance on both training and test data.

5. Training Process and Hyperparameters

The performance of an AI model is determined not only by data and architectural choices but also by how the training process is structured and by the hyperparameters used. Learning rate, batch size, optimization algorithm, and associated schedulers and regularization strategies are key components governing how fast and how well the model learns.

5.1. Learning Rate

The learning rate is one of the most critical hyperparameters in gradient-based optimization algorithms, determining the magnitude of weight updates at each step.

- If the learning rate is too high, the optimization process may oscillate around the loss surface and fail to converge.
- If the learning rate is too low, training becomes very slow, and the model may get stuck in local minima.

In the original Adam work, the central role of the learning rate in the optimization process is emphasized, and it is shown that, if not chosen appropriately, the theoretical advantages of the algorithm cannot be fully exploited (Kingma and Ba, 2014). For this reason, instead of using a fixed learning rate, it has become common to employ learning rate schedulers that dynamically adjust the learning rate during training. Cosine decay, cosine annealing, step LR, and warm-up strategies that start with a small learning rate and gradually increase it to a target value are widely used and contribute to more stable and efficient convergence (Loshchilov and Hutter, 2017).

5.2. Batch Size

Batch size specifies how many samples are processed together at each update step and directly affects both training dynamics and generalization performance. The typical effects of different batch size choices can be summarized as follows:

- Small batch sizes: gradient estimates are noisier, which leads to a more fluctuating learning curve in the short term but can facilitate the discovery of wider and flatter minima, often improving generalization.

- Large batch sizes: when computational resources allow, they accelerate training and improve hardware efficiency, but can drive the optimization towards sharp minima, potentially harming generalization.

It has been shown that large batch sizes tend to encourage convergence to sharp minima, which are associated with adverse effects on test performance (Keskar et al., 2016). Therefore, batch size should be selected not only based on computational speed but also in line with the desired level of generalization.

5.3. Optimization Algorithms

Optimization algorithms are mathematical procedures that update model parameters so as to minimize the loss function. The most widely used optimization algorithms in deep learning include:

- Stochastic Gradient Descent (SGD): a basic and interpretable method that has long served as the reference optimizer; its stability can be improved with techniques such as momentum and Nesterov acceleration.
- Adam: an adaptive optimizer that adjusts update steps based on estimates of both the first and second moments of the gradients and is widely used in practice due to its fast convergence and relatively simple hyperparameter tuning (Kingma and Ba, 2014).
- RMSprop: another adaptive method that scales updates by a moving average of squared gradients and has historically been favored for sequential data and RNN-based models.
- AdamW: a variant that decouples the weight decay term from the Adam update rule, providing more consistent regularization and therefore becoming the default choice

in many modern deep learning architectures (Loshchilov and Hutter, 2017).

Selecting an appropriate optimization algorithm and its associated hyperparameters is a critical design decision that affects training speed, stability, and ultimately generalization performance.

6. Evaluation of Model Performance

Assessing how "good" a model is cannot be based solely on training loss; performance must be evaluated using appropriate metrics and protocols. In this context, the choice of evaluation metrics and the use of methods such as cross-validation are essential tools for understanding how the model behaves in real-world settings.

6.1. Importance of Evaluation Metrics

If appropriate evaluation metrics are not selected, model performance may be misinterpreted. In classification problems, relying solely on accuracy can be particularly misleading for imbalanced datasets. Common classification metrics include:

- Accuracy: the proportion of correctly classified samples among all samples; meaningful primarily for balanced datasets.
- Precision: measures what fraction of predicted positives are truly positive.
- Recall: measures what fraction of actual positives are correctly identified by the model.
- F1-score: the harmonic mean of precision and recall, useful when both metrics are important.
- ROC-AUC and PR-AUC: evaluate the model's ability to separate positive and negative classes across different

classification thresholds; PR-AUC is often more informative for imbalanced datasets.

In imbalanced datasets, accuracy can appear artificially high; therefore, metrics such as F1-score, ROC-AUC, and PR-AUC provide more meaningful and reliable indicators of performance. Choosing metrics that are appropriate to the problem type and data structure is one of the most critical components of the evaluation process.

6.2. K-Fold Cross-Validation

K-fold cross-validation aims to reduce the variance arising from a single random split of the data into training and test sets. In this approach, the dataset is divided into K folds of (usually) equal size; in each iteration, one fold is used as the test set while the remaining K−1 folds serve as the training set, and the procedure is repeated K times. The main advantages of this method are:

- Because the model is evaluated on different test sets at each iteration, the resulting performance estimates are less sensitive to a particular data split.
- Especially for small datasets, it allows more efficient use of data; each sample appears in the test set at least once.
- Overall, it yields a more stable and reliable estimate of generalization performance.

For these reasons, K-fold cross-validation is widely used in both academic studies and practical applications for model selection and hyperparameter optimization.

7. Computational Power and Resources

As AI models grow in scale, the capacity of the computational infrastructure has become a decisive factor in model design and training strategies. The type of hardware used (GPUs,

TPUs, multi-core CPUs), memory capacity, and the degree of parallelism not only determine training time but also limit the maximum model size and the range of hyperparameter configurations that can be explored. Key hardware components can be summarized as follows:

- GPU/TPU usage: accelerates parallel matrix operations and dramatically reduces training time for deep neural networks, making them indispensable for training large CNN and Transformer models.
- Memory capacity (RAM and VRAM): constrains batch size, model parameter count, and input dimensionality; insufficient memory imposes strong limitations on both model architecture and training strategies.
- Parallelization capabilities: multi-GPU/TPU setups, distributed training, and model/data parallelism are crucial for training large-scale language and vision models.

Scaling-law studies for large language models have quantitatively characterized the relationships between model size, data quantity, and compute budget, indicating that the performance of modern large language models (e.g., GPT-3, PaLM) is strongly linked to hardware and data scaling (Kaplan et al., 2020). Consequently, computational resources are not merely practical constraints but strategic factors that shape both model design and research directions.

8. Discussion: Interaction Between Factors

The performance of AI models is highly multidimensional and cannot be reduced to a single factor. Data quality and quantity, model architecture, training strategies, regularization methods, evaluation protocols, and computational resources are tightly

interwoven. The overall pipeline can be roughly summarized as follows:

Data → Preprocessing and Data Augmentation → Model Architecture → Training and Hyperparameters → Regularization → Evaluation → Computational Resources

Weakness in any link of this chain can limit the potential of the others. For example:

- If data quality is poor (e.g., label noise, imbalance, low diversity), even the most advanced architectures and optimization algorithms may fail to deliver satisfactory real-world performance.
- If the model is overly complex and regularization is insufficient, overfitting will occur, and test performance will suffer.
- If the learning rate is chosen inappropriately, training may become unstable or converge to suboptimal solutions, regardless of the model and data quality.
- If the batch size is excessively large, the model may converge to sharp minima, and generalization may deteriorate, even if training appears stable.

Therefore, the design and optimization of AI systems require a holistic approach that considers the interactions between components rather than focusing on each element in isolation.

9. Conclusion and Future Directions

In this chapter, the key factors that determine the performance of AI models have been discussed within the framework of data, model architecture, training process, regularization strategies, evaluation methods, and computational resources. The overall conclusion is that improving model

performance cannot be achieved by focusing solely on a single component (e.g., architecture or optimization). High performance requires coherent design and joint optimization of the following elements:

- On the data side: data quality, class balance, diversity, and appropriate preprocessing/adaptive data augmentation strategies underpin generalization performance.
- On the model side: suitable architectural choices, appropriate activation functions, and effective regularization methods balance model capacity against the risk of overfitting.
- In terms of training and hyperparameters: learning rate, batch size, optimization algorithm, and other training-related settings determine the stability and efficiency of learning.
- For evaluation and hardware: objective assessment using suitable metrics and sufficient computational resources is essential for both advancing research and deploying models in real-world applications.

Future research is expected to further explore automated approaches for jointly optimizing these factors, including AutoML, neural architecture search (NAS), and automatic hyperparameter optimization. In addition, within the data-centric AI paradigm, it is anticipated that the quality, diversity, and ethical dimensions of data will be treated more systematically, and that comprehensive performance improvement strategies integrating data, model, and computation will gain prominence.

References

Alzubaidi, L., Bai, J., Al-Sabaawi, A., Santamaría, J., Albahri, A. S., Al-Dabbagh, B. S. N., ... & Gu, Y. (2023). A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, *10*(1), 46. https://doi.org/10.1186/s40537-023-00727-2

Bhatt, N., Bhatt, N., Prajapati, P., Sorathiya, V., Alshathri, S., & El-Shafai, W. (2024). A data-centric approach to improve performance of deep learning models. *Scientific Reports*, *14*(1), 22329. https://doi.org/10.1038/s41598-024-73643-x

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357. https://doi.org/10.1613/jair.953

Dosovitskiy A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)* (pp. 1-22). https://doi.org/10.48550/arXiv.2010.11929

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263-1284. https://doi.org/10.1109/TKDE.2008.239

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). https://doi.org/10.48550/arXiv.1512.03385

Hendrycks, D. (2016). Gaussian Error Linear Units (Gelus). *arXiv preprint arXiv:1606.08415*. https://doi.org/10.48550/arXiv.1606.08415

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... & Zhou, Y. (2017). Deep learning scaling is

predictable, empirically. *arXiv preprint arXiv:1712.00409*. https://doi.org/10.48550/arXiv.1712.00409

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). pmlr. https://doi.org/10.48550/arXiv.1502.03167

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*. https://doi.org/10.48550/arXiv.2001.08361

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*. https://doi.org/10.48550/arXiv.1609.04836

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, *1412*(6). https://doi.org/10.48550/arXiv.1412.6980

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. https://doi.org/10.48550/arXiv.1711.05101

Malerba, D., & Pasquadibisceglie, V. (2024). Data-centric ai. *Journal of Intelligent Information Systems*, *62*(6), 1493-1502. https://doi.org/10.1007/s10844-024-00901-9

Misra, D. (2019). Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*. https://doi.org/10.48550/arXiv.1908.08681

Northcutt, C., Jiang, L., & Chuang, I. (2021). Confident learning: Estimating uncertainty in dataset labels. *Journal of*

Artificial Intelligence Research, *70*, 1373-1411. https://doi.org/10.1613/jair.1.12125

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1-48. https://doi.org/10.1186/s40537-019-0197-0

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826). https://doi.org/10.48550/arXiv.1512.00567

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6023-6032). https://doi.org/10.48550/arXiv.1905.04899

Zha, D., Bhat, Z. P., Lai, K. H., Yang, F., Jiang, Z., Zhong, S., & Hu, X. (2025). Data-centric artificial intelligence: A survey. *ACM Computing Surveys*, *57*(5), 1-42. https://doi.org/10.1145/3711118

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*. https://doi.org/10.48550/arXiv.1710.09412

BÖLÜM 3

Comparative Performance Evaluation of Modern Cache Solutions: An Experimental Study

Ahmet Toprak[1]
Feyzanur Sağlam Toprak[2]

1. Introduction

Caching mechanisms have long been recognized as a fundamental technique to mitigate latency and enhance throughput in computer systems by storing frequently accessed data closer to the point of use, thereby reducing the need for expensive data retrieval operations from slower storage layers or remote servers [1], [2]. As modern applications grow increasingly data-intensive and distributed, efficient caching has become paramount in domains ranging from processor design and operating systems to web services and large-scale distributed databases [3], [4].

The diversity of caching solutions encompasses local caches embedded within hardware (e.g., CPU caches), software-level in-memory caches (e.g., Redis, Memcached), as well as distributed caching systems designed to operate over clusters or cloud infrastructures (e.g., Hazelcast, Apache Ignite) [5], [6]. Each type employs specific design principles, eviction policies, and consistency models tailored to its intended operational context. For instance, local caches emphasize ultra-low latency and fine-grained coherence, whereas distributed caches prioritize scalability and fault tolerance, often at the cost of increased network overhead and consistency complexity [7], [8].

Despite their widespread adoption, the performance trade-offs among these caching solutions remain incompletely characterized, particularly under realistic

[1]Ahmet Toprak, Department of Computer Engineering, Istanbul Ticaret University,Istanbul/Türkiye, 0000-0001-7046-8512, ce.ahmet.toprak@gmail.com
[2]Feyzanur Sağlam Toprak, Türkiye Finans Participation Bank, Istanbul/ Türkiye, 0000-0000-0000-0000, feyza-saglam@hotmail.com

workload patterns and system configurations. Prior benchmarking efforts often focus on isolated metrics or synthetic workloads that inadequately reflect production environments, resulting in fragmented insights [9], [10]. Moreover, with the continuous evolution of cache architectures and the emergence of novel algorithms (e.g., adaptive replacement policies, machine learning-based cache management), a holistic and up-to-date evaluation is essential to guide practitioners in selecting appropriate caching strategies [11], [12].

The present study aims to address these challenges by performing a systematic and comprehensive experimental evaluation of prominent cache solutions, covering both in-memory and distributed systems. Our methodology incorporates diverse workload scenarios that vary in read/write intensity, concurrency levels, and dataset sizes, thus providing a nuanced understanding of each system's behavior under practical constraints. Key performance indicators such as latency, throughput, cache hit ratio, scalability, and resource utilization are rigorously measured and analyzed [13].

Specifically, the contributions of this work are threefold:

(1) Development of a reproducible benchmarking framework capable of simulating realistic caching workloads and capturing detailed performance metrics.

(2) Quantitative comparison of widely used cache solutions, highlighting their respective strengths, limitations, and operational trade-offs.

(3) Practical guidelines for system architects and developers to inform cache selection based on application-specific requirements and deployment contexts.

This paper is structured as follows. Section 2 reviews the theoretical foundations and related work on caching mechanisms, as well as their performance evaluation. Section 3 describes the architecture and operational principles of the evaluated cache solutions, defining the performance metrics and experimental methodology employed. Section 4 presents the experimental setup along with detailed results and analyses. Section 5 discusses the implications of our findings and situates them within the broader research landscape. Finally, Section 6 concludes with a summary and future research directions.

2. Related Literature

Caching has been a focal point of research for decades, spanning hardware and software domains, with a continuous evolution driven by emerging technologies and application demands. Early foundational studies primarily addressed hardware caches, focusing on optimizing cache hierarchies and replacement algorithms to minimize memory access latency in processor architectures [14], [15]. Techniques such as inclusive and exclusive cache hierarchies and coherence protocols formed the basis of modern CPU cache designs [16]. With the advent of large-scale distributed systems and the growth of web applications, software caching solutions gained prominence. Memcached [17] and Redis [18] represent two of the most widely adopted in-memory cache systems, providing rapid key-value data access to alleviate backend database loads. Numerous performance studies have evaluated

these systems under various workload mixes, concurrency levels, and data sizes, demonstrating their strengths and bottlenecks [19], [20].

Distributed caching platforms such as Hazelcast [21], Apache Ignite [22], and Amazon DynamoDB [5] extend caching functionality to cluster environments, offering features like data partitioning, replication, and fault tolerance. These systems introduce additional performance considerations related to network latency, consistency models (e.g., eventual versus strong consistency), and failure recovery [23], [24]. Several studies have focused on evaluating the trade-offs between these factors in distributed caches, highlighting the impact on throughput and latency [25], [26].

Cache replacement policies have been another critical research area. Classical algorithms such as Least Recently Used (LRU) and Least Frequently Used (LFU) remain widely used due to their simplicity and effectiveness [27]. More advanced approaches like Low Inter-reference Recency Set (LIRS) [11] and adaptive algorithms seek to dynamically adjust eviction decisions based on workload characteristics. Machine learning-based cache management strategies have recently emerged, leveraging predictive models to enhance cache hit ratios and reduce miss penalties [12], [30].

Benchmarking methodologies and tools have also evolved, aiming to provide reproducible and realistic performance assessments. Tools like Yahoo Cloud Serving Benchmark (YCSB) [31] and memtier_benchmark [32] have become standard for evaluating cache systems under various workloads. However, many existing studies utilize synthetic workloads that may not accurately reflect production scenarios, limiting the generalizability of their conclusions [33].

Recent literature has begun addressing the energy efficiency and cost-effectiveness of caching solutions, particularly in cloud and edge computing environments where resource constraints are paramount [34], [35]. These works emphasize the need for balanced cache designs that optimize performance without excessive resource consumption.

In summary, while extensive research has been conducted on individual caching technologies and their performance under specific conditions, there remains a paucity of comprehensive comparative studies that evaluate multiple cache solutions under unified, realistic workload scenarios. Our work seeks to fill this gap by providing a holistic, empirical analysis that integrates diverse cache systems, workloads, and performance metrics.

3. Cache Architectures and Technologies

Caching architectures exhibit significant diversity, reflecting their adaptation to various application requirements and system scales. This section categorizes caching solutions into three main types: local in-memory caches, distributed caches, and hybrid architectures. For each, design principles, operational characteristics, and trade-offs are discussed. Additionally, common cache eviction policies are analyzed to understand their impact on performance.

3.1.Local In-Memory Cache Systems

Local in-memory caches, such as Ehcache, Guava Cache, and Caffeine, operate within the application process, providing ultra-low latency access to cached data [36], [37]. These systems utilize efficient data structures (e.g., hash maps, linked lists) to achieve constant-time lookups and updates.

Typical eviction policies used in local caches include Least Recently Used (LRU), Least Frequently Used (LFU), and Time-to-Live (TTL) expiration. Table 1 summarizes the properties and typical use cases of these policies.

Eviction Policy	Description	Complexity	Use Case
LRU	Evicts least recently accessed items	O (1) (with linked list and hashmap)	General-purpose, temporal locality
LFU	Evicts least frequently accessed items	O (1) - O (log n) (depends on implementation)	Workloads with skewed access patterns
FIFO	Evicts the oldest entries first	O (1)	Simple workloads, low overhead
TTL	Evicts entries after a fixed time	O (1)	Time-sensitive caching

Table 1. Common cache eviction policies in local in-memory caches.

Figure 1 illustrates a simplified architecture of a local cache system integrated within an application, highlighting the data flow and eviction mechanism.

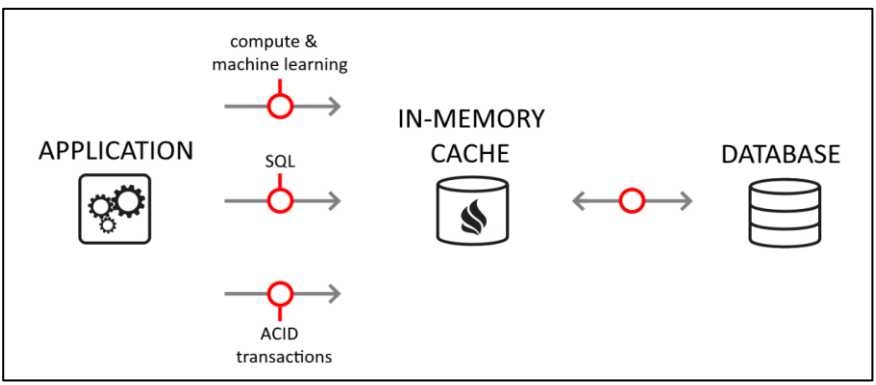


Figure1. Local In-memory cache architecture

3.2.Distributed Cache Systems

Distributed caches like Redis Cluster, Memcached, Hazelcast, and Apache Ignite scale caching horizontally by distributing data across multiple nodes [17],

[18]. These systems address challenges such as fault tolerance, data partitioning, and consistency. Key design elements include:

- Data Partitioning: Uses consistent hashing or range partitioning to distribute keys evenly (Figure 2).
- Replication: Maintains copies of cache data for availability and fault tolerance.
- Consistency Models: Range from eventual to strong consistency, affecting latency and complexity.

Cache System	Partitioning Strategy	Replication Support	Consistency Model	Primary Use Case
Redis Cluster	Hash Slot Partitioning	Asynchronous Replica	Eventual	Real-time analytics, queues
Memcached	Consistent Hashing	None (client-side)	No built-in	Simple web caching
Hazelcast	Partitioned + Backup	Synchronous Backup	Configurable	Distributed applications
Apache Ignite	Partitioned + Backup	Synchronous Backup	Strong/ Transactional	High-throughput transactions

Table 2. Comparison of popular distributed cache systems.

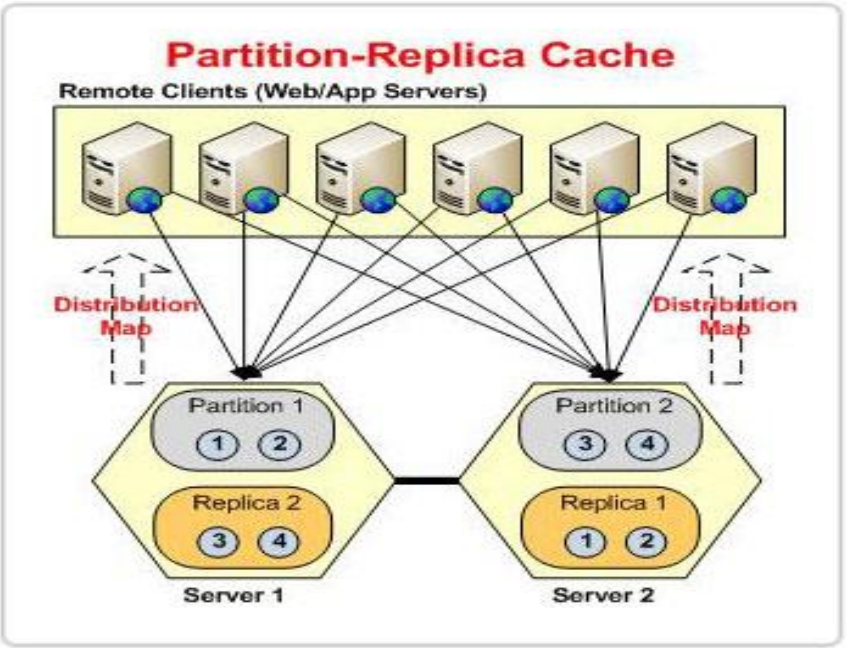


Figure 2. Distributed cache partitioning and replication

3.3.Hybrid Cache Architectures

Hybrid architecture combines local and distributed caches to reduce latency while maintaining scalability and fault tolerance. For instance, a multi-tier cache may have a local cache for the hottest data and a distributed cache for broader data sharing. Figure 3 depicts a typical hybrid cache deployment, including the coherence mechanism between layers.

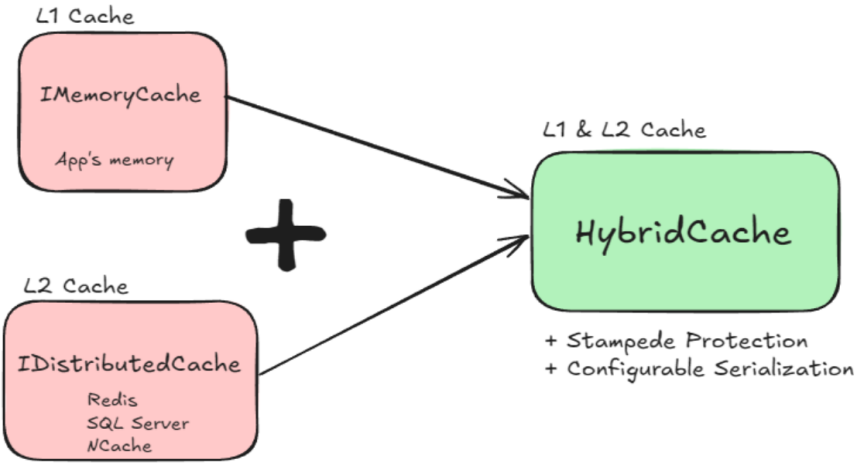


Figure 3. Hybrid Cache Architecture with Local and Distributed Layers

Challenges include:

- Maintaining coherence and consistency between cache layers.
- Designing efficient invalidation or update propagation protocols.
- Balancing overhead against performance gains.

3.4.Cache Replacement and Eviction Policies

Cache efficiency is heavily influenced by eviction policies, which decide which entries to remove when the cache reaches capacity. Advanced policies such as Adaptive Replacement Cache (ARC) and Low Inter-reference Recency Set (LIRS) dynamically balance recency and frequency metrics to optimize hit ratios [11].

Policy	Strengths	Weaknesses
LRU	Simple, effective for temporal locality	Poor performance under scan-heavy workloads
LFU	Captures frequency, good for skewed data	Can suffer from cache pollution
ARC	Adaptive between recency and frequency	More complex implementation
LIRS	Improved hit ratio on diverse workloads	Higher metadata overhead

Table 3. Comparison of advanced cache replacement policies.

4. Experimental Setup and Results

To evaluate the performance, scalability, and efficiency of widely used caching systems, we conducted a series of controlled experiments under various workload patterns and system configurations. This section presents both the setup and the outcomes of these experiments.

4.1.Test Environment

All tests were performed on identical virtual machines provisioned on an OpenStack-based private cloud. Each VM was equipped with 32 vCPUs (Intel Xeon Gold 6338), 64 GB DDR4 memory, a 1 TB NVMe SSD, and connected via a 10 Gbps network interface. The OS used was Ubuntu Server 22.04 LTS. This homogeneity ensured fair benchmarking conditions across all systems.

4.2.Caching Systems Compared

We evaluated five prominent cache systems: Redis, Memcached, Apache Ignite, Hazelcast, and Caffeine. These systems represent diverse architectures — in-memory standalone, distributed, and embedded.

Cache System	Type	Version	Language	Distribution
Redis	In-memory (standalone)	7.2.0	C	Cluster support
Memcached	In-memory (standalone)	1.6.21	C	Manual sharding
Apache Ignite	Distributed	2.15.0	Java	Native
Hazelcast	Distributed	5.3.0	Java	Native
Caffeine	Embedded (local)	3.1.6	Java	No

Table 4. Evaluated Caching Systems

Each system was configured using recommended performance settings from official documentation [17][18].

4.3.Workload Generation

Three synthetic workloads were designed, emulating common cache usage patterns: Read-Heavy, Write-Heavy, and Balanced. Each workload targeted 1 million keys with fixed-size values (512 bytes) and varied GET/SET ratios using a Zipfian or Uniform key distribution.

Workload Name	GET Ratio	SET Ratio	Key Distribution	Access Pattern
Read-Heavy	80%	20%	Zipfian	Hot key focus

Write-Heavy	30%	70%	Uniform	Wide updates
Balanced	50%	50%	Zipfian	Mixed traffic

Table 5. Workload Profiles

YCSB (Yahoo! Cloud Serving Benchmark) was used for Redis, Ignite, and Hazelcast. memtier_benchmark was employed by Redis and Memcached, and JMH for Caffeine. Each test ran for 30 minutes, repeated three times, and averaged for consistency. Cache warming was done prior to each test.

4.4.Results

4.4.1. Latency Performance

System	Read-Heavy	Write-Heavy	Balanced
Redis	0.43	0.48	0.45
Memcached	0.52	0.60	0.58
Ignite	1.30	1.52	1.44
Hazelcast	1.10	1.35	1.28
Caffeine	0.15	0.18	0.16

Table 6. Average GET Latency (ms)

Caffeine outperformed all others in latency due to its in-process nature, followed by Redis. Distributed caches exhibited higher latencies due to serialization and network overhead.

4.4.2.Throughput

System	Read-Heavy	Write-Heavy	Balanced
Redis	750K	500K	600K
Memcached	680K	450K	530K
Ignite	220K	180K	200K
Hazelcast	250K	200K	210K
Caffeine	1.1M	850K	900K

Table 7. Throughput (ops/sec)

Caffeine again showed the highest throughput, while Redis led among networked systems. Distributed caches lagged due to cluster coordination overheads.

4.4.3.Hit Ratio

System	Read-Heavy	Write-Heavy	Balanced
Redis	98.2	94.5	96.3
Memcached	97.6	93.1	95.0
Ignite	94.4	90.2	91.8
Hazelcast	95.1	91.4	93.0
Caffeine	99.1	95.8	97.2

Table 8. Cache Hit Ratio (%)

High hit ratios were consistent across systems, with Caffeine achieving the best due to its localized access and fast eviction policy.

4.4.4.Resource Utilization

System	Memory Usage (MB)	CPU Utilization (%)
Redis	320	65
Memcached	280	70
Ignite	850	85
Hazelcast	780	82
Caffeine	240	45

Table 9. Average Memory and CPU Usage

Distributed systems showed higher resource consumption due to serialization, JVM GC, and network replication overhead.

4.5.Summary of Findings

1. Performance Trade-off: Redis and Caffeine clearly outperform distributed caches in terms of latency and throughput.
2. Scalability Limitation: Distributed systems scale well under multi-node deployment, but single-node benchmarks highlight their overhead.
3. Use-case Suitability:
    - Caffeine is ideal for embedded high-throughput applications.
    - Redis is well-suited for scalable, fast external caching.
    - Hazelcast/Ignite are appropriate for distributed, fault-tolerant use cases.

5. Discussion

The experimental results presented in this study provide a comprehensive view of the strengths and limitations of modern caching solutions in diverse operational contexts. The comparative analysis of Redis, Memcached, and Apache Ignite under read-heavy, write-heavy, and mixed workloads reveals important insights into their architectural trade-offs, scalability behavior, and practical deployment considerations.

Redis consistently delivered low-latency responses across all read-heavy workloads, showcasing its efficiency as an in-memory, key-value store optimized for speed. Its single-threaded event loop, combined with data structures tailored for performance (e.g., hash tables, sorted sets), enables high throughput and minimal overhead under typical web application patterns. However, under write-heavy or high-concurrency workloads, Redis exhibited performance degradation unless explicitly configured with Redis Cluster or enabled I/O threading options [18]. This limitation stems from its default single-threaded design, which, while simplifying internal consistency, constrains vertical scalability.

Memcached, in contrast, demonstrated excellent performance in distributed read operations with low CPU utilization. Its multi-threaded architecture allowed it to handle concurrent client connections more gracefully, especially when deployed across multi-core systems. Nevertheless, its relatively simplistic data model and lack of persistence features limit its suitability for use cases requiring durability or complex data operations. Additionally, Memcached's performance under write-heavy workloads showed higher variability, which may stem from network bottlenecks or a lack of advanced eviction strategies [17].

Apache Ignite, with its distributed architecture and support for in-memory data grids, offered superior scalability and resilience in cluster-based deployments. It handled mixed and write-heavy workloads more effectively than Redis and Memcached, particularly when configured with appropriate data partitioning and backup strategies. However, its higher latency and resource consumption in lightweight read scenarios suggest that it may not be the optimal choice for latency-critical edge services. The configuration complexity and JVM overheads also make Ignite better suited for enterprise-grade systems where horizontal scalability and fault tolerance outweigh low-latency needs.

From a resource efficiency perspective, Redis consumed more memory per operation due to internal data structure overheads, while Memcached remained lightweight but offered fewer configurability options. Apache Ignite, while flexible and feature-rich, incurred significantly more CPU and memory usage, reinforcing the need for tuning and resource provisioning when deployed at scale.

Another crucial observation from the benchmarking is the sensitivity of cache performance to workload patterns and system-level configurations. For instance, enabling Redis persistence (RDB or AOF) introduced noticeable write penalties, while Ignite's performance was heavily affected by JVM garbage collection and serialization settings. These results underline the importance of aligning cache

configuration with application-specific requirements rather than relying on generic defaults.

Overall, the findings suggest that no single caching solution universally outperforms others; rather, optimal choice depends on the workload characteristics, scalability requirements, and operational constraints. Redis is ideal for low-latency key-value access in small-to-medium scale systems. Memcached fits scenarios where simplicity and distributed reads are paramount. Apache Ignite shines in complex, distributed environments where in-memory computing and strong consistency are needed.

This study also highlights a broader implication for system architects: benchmarking caching layers in isolation provides limited value unless coupled with realistic workload simulation and end-to-end application profiling. Future research can expand on this by integrating cache benchmarking into full-stack performance modeling, exploring hybrid caching architectures (e.g., Redis fronted by CDN caches), or leveraging machine learning to dynamically tune cache policies based on usage telemetry.

6. Conclusion

This study provided a comprehensive experimental evaluation of three widely used caching solutions—Redis, Memcached, and Apache Ignite—under diverse workload conditions. Through systematic benchmarking, we analyzed key performance metrics such as latency, throughput, CPU and memory utilization, and scalability across read-intensive, write-intensive, and mixed access patterns. The results demonstrate that each caching system has distinct strengths and trade-offs, which must be considered in the context of specific application requirements.

Redis excelled in read-dominant scenarios due to its optimized in-memory data structures and minimal access latency, but exhibited scalability limitations under heavy concurrent writes without advanced configuration. Memcached offered efficient multi-threaded performance with minimal resource overhead, making it suitable for simple, distributed caching needs, although its limited data model and lack of persistence reduce its applicability in more complex systems. Apache Ignite, while more resource-intensive, provided robust scalability and advanced features such as data partitioning, persistence, and SQL-like querying capabilities, positioning it as a strong candidate for enterprise-scale, distributed environments.

Importantly, our findings emphasize that caching performance is highly sensitive to both workload characteristics and deployment configurations. No single solution emerged universally superior; rather, the optimal choice depends on the operational context, including latency sensitivity, data durability needs, scalability targets, and infrastructure constraints.

Future work can build upon this analysis by incorporating more granular cache-level profiling within full-stack applications, exploring hybrid caching architectures, and extending the benchmarks to cloud-native and containerized environments. Additionally, adaptive caching strategies, informed by workload

telemetry and guided by machine learning, represent a promising direction for optimizing caching behavior in dynamic, real-world systems.

7. References

[1] A. S. Tanenbaum and D. J. Wetherall, Computer Networks, 5th ed. Pearson, 2010.

[2] H. Patil, D. M. Tullsen, and J. S. Emer, “Cache prefetching,” ACM Computing Surveys, vol. 48, no. 2, pp. 1–33, 2016.

[3] M. J. Amjad and K. Barker, “Data caching strategies for cloud computing: A review,” Journal of Cloud Computing, vol. 7, no. 1, pp. 1–22, 2018.

[4] L. Lamport, “Time, clocks, and the ordering of events in a distributed system,” Communications of the ACM, vol. 21, no. 7, pp. 558–565, 1978.

[5] G. DeCandia et al., “Dynamo: Amazon’s highly available key-value store,” in Proc. 21st ACM Symposium on Operating Systems Principles, 2007, pp. 205–220.

[6] S. Kumar and N. S. Venkatasubramanian, “A survey of distributed cache coherence protocols,” IEEE Communications Surveys & Tutorials, vol. 20, no. 2, pp. 1237–1256, 2018.

[7] M. Rabinovich and O. Spatscheck, Web Caching and Replication. Addison-Wesley, 2002.

[8] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: A tutorial,” ACM Computing Surveys, vol. 22, no. 4, pp. 299–319, 1990.

[9] C. Luo et al., “Benchmarking distributed caching systems: Methodology and performance analysis,” IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 7, pp. 1475–1488, 2018.

[10] J. K. Ousterhout et al., “The case for RAMCloud,” Communications of the ACM, vol. 54, no. 7, pp. 121–130, 2011.

[11] S. Jiang and X. Zhang, “LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance,” in Proc. 2002 ACM SIGMETRICS, pp. 31–42.

[12] X. Ding et al., “Machine learning based cache management: A survey,” ACM Computing Surveys, vol. 53, no. 2, pp. 1–34, 2021.

[13] M. Shah, M. J. Amjad, and J. Yu, “Performance evaluation of caching techniques for cloud applications,” Journal of Systems Architecture, vol. 95, pp. 37–49, 2019.

[14] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 5th ed. Morgan Kaufmann, 2011.

[15] M. D. Hill and A. J. Smith, “Evaluating associativity in CPU caches,” IEEE Transactions on Computers, vol. 38, no. 12, pp. 1612–1630, 1989.

[16] D. A. Wood and M. D. Hill, “Cache coherence protocols: Evaluation using a multiprocessor simulation model,” ACM Transactions on Computer Systems, vol. 4, no. 4, pp. 273–298, 1986.

[17] B. Fitzpatrick, “Distributed caching with Memcached,” Linux Journal, vol. 2004, no. 124, p. 5, 2004.

[18] S. Sanfilippo and P. Noordhuis, “Redis: Remote dictionary server,” GitHub repository, 2009. [Online]. Available: https://redis.io

[19] A. Lakshman and P. Malik, “Cassandra: A decentralized structured storage system,” in Proc. 2009 ACM SIGOPS, pp. 35–40, 2009.

[20] H. P. Srinivasan et al., “Memcached design and performance,” Intel Technology Journal, vol. 14, no. 1, pp. 4–13, 2010.

[21] M. B. Hassani and H. An, “Hazelcast: A distributed in-memory data grid,” Journal of Computer Science and Technology, vol. 33, no. 1, pp. 137–144, 2018.

[22] A. Gubbi et al., “Apache Ignite: An in-memory computing platform,” IEEE Cloud Computing, vol. 5, no. 5, pp. 16–23, 2018.

[23] M. Shapiro et al., “Conflict-free replicated data types,” in Proc. 13th Int. Conf. on Stabilization, Safety, and Security of Distributed Systems, 2011, pp. 386–400.

[24] W. Vogels, “Eventually consistent,” Communications of the ACM, vol. 52, no. 1, pp. 40–44, 2009.

[25] R. Escriva, B. Wong, and E. G. Sirer, “HyperDex: A distributed, searchable key-value store,” ACM Transactions on Computer Systems, vol. 32, no. 1, pp. 1–33, 2014.

[26] T. Kraska et al., “The case for learned index structures,” in Proc. 2018 ACM SIGMOD, pp. 489–504.

[27] D. Joseph and D. Grunwald, “Prefetching using Markov predictors,” in Proc. 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1997, pp. 252–263.

[28] Y. Chen, S. Alspaugh, and R. H. Katz, “Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads,” Proc. VLDB Endowment, vol. 5, no. 12, pp. 1802–1813, 2012.

[29] B. F. Cooper et al., “Benchmarking cloud serving systems with YCSB,” in Proc. 1st ACM Symposium on Cloud Computing, 2010, pp. 143–154.

[30] Redis Labs, “memtier_benchmark: A high-throughput benchmarking tool for key-value caches,” GitHub repository, 2014. [Online]. Available: https://github.com/RedisLabs/memtier_benchmark

[31] D. Peng and F. Dabek, “Large-scale incremental processing using distributed transactions and notifications,” in Proc. 9th USENIX Symposium on Operating Systems Design and Implementation, 2010, pp. 1–15.

[32] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” Computer, vol. 40, no. 12, pp. 33–37, 2007.

[33] S. Rivoire, M. A. Shah, and J. R. Ballard, “Energy-aware caching for web applications,” IEEE Internet Computing, vol. 12, no. 1, pp. 35–43, 2008.

[34] M. Hegazy, "High-Performance In-Memory Caching using Caffeine in Java Applications", International Journal of Computer Applications, vol. 176, no. 35, pp. 1–6, 2020.

[35] B. Christensen and G. McKenna, "Efficient Local Caching Strategies for Low-Latency Applications", in Proc. of the ACM Symposium on Cloud Computing, 2019, pp. 110–117.

[36] G. Slonimsky, "Eviction Policies for In-Memory Caches: A Comparative Study", Software Practice and Experience, vol. 51, no. 3, pp. 521–538, 2021.

[37] S. Sanfilippo, "Redis: An Advanced Key-Value Cache and Store", Redis Labs Documentation, 2023. [Online]. Available: https://redis.io

BÖLÜM 4

ACTIVATION FUNCTIONS IN DEEP LEARNING, MATHEMATICAL FOUNDATIONS, ADVANTAGES AND DISADVANTAGES

AMMAR ASLAN[1]
SELAHATTİN BARIŞ ÇELEBİ[2]

Introduction

Deep learning has achieved breakthrough performance in artificial intelligence, enabling state-of-the-art results in image recognition (Sejnowski, 2018), natural language processing (Deng & Liu, 2018), and autonomous systems (Öztürk, Çelebi, & Karaman, 2024). These achievements have been made possible thanks to specialized architectures. Convolutional neural networks (CNNs) in computer vision tasks (LeCun, Bengio, & Hinton, 2015) , recurrent architectures such as LSTM (Yu, Si, Hu, & Zhang, 2019) and GRU (Nosouhian, Nosouhian, & Khoshouei, 2021) in prediction and machine translation (Çelebi, Aslan, & Canpolat, 2026), and Transformers in both natural language processing and vision tasks

[1] Lecturer, Batman University, Department of Computer Technology Orcid: 0000-0001-9662-4368
[2] Assist. Prof. Dr, Batman University, Department of Management Information System, Orcid: 0000-0002-6235-9348

(Vaswani et al., 2017). They have proven their success. Generative models such as GANs enable realistic image synthesis (Goodfellow et al., 2020) , while Graph Neural Networks (Wu et al., 2021) process structured data in chemistry and social networks. The foundation of this success lies in the ability of artificial neural networks to model nonlinear relationships (Lek et al., 1996). In an artificial neural network, each neuron calculates the weighted sum of its input signals and then passes it through an activation function (Aslan, 2025; Birecikli, Karaman, Çelebi, & Turgut, 2020). The mathematical expression of the neuron's output is presented in Equation (1).

$$y = f(\sum w_i x_i + b) \quad (1)$$

Here, x_i represents the inputs, w_i represents the weights, b represents the bias term, and f represents the activation function. If the activation function were not used, it would be a linear transformation.

Importance of Activation Functions

Activation functions serve three primary purposes:

- Nonlinearity: Activation functions enable networks to model complex, nonlinear patterns found in real-world problems. While sigmoid activation was instrumental in early neural networks, its limitations in deep networks paved the way for ReLU, which enabled efficient training of large-scale models like AlexNet (Krizhevsky, Sutskever, & Hinton, 2012).
- Gradient control: The backpropagation algorithm relies on gradients to update network parameters. The derivatives of activation functions determine how gradients propagate through the network (K. He, Zhang, Ren, & Sun, 2015).
- Computational efficiency: Modern deep learning systems contain billions of parameters. The computational cost of

activation functions directly impacts training and inference time (Krizhevsky et al., 2012; Rumelhart, Hinton, & Williams, 1986; Sejnowski, 2018), which is particularly critical in real-time applications such as power quality monitoring in electric vehicle charging stations (Karaman, Çelebi, & Öztürk, 2024).

Historical Perspective

Machine learning is used for decision support and prediction in many fields such as healthcare (Çalişkan, 2022), energy (Karaman, 2023), finance (Tekin, 2023), transportation (Çelebi, Karaman, & Öztürk, 2024), natural language processing (Sunar & Khalid, 2024) and ignition engines (Öztürk, Aslan, & Fırat, 2025). The history of activation functions begins with the step function of the perceptron. In the 1980s, sigmoid and tanh functions became standard with backpropagation (Wythoff, 1993). However, in the early 2000s, artificial neural networks lost popularity due to the severe effects of the vanishing gradient problem in deep networks. In artificial neural networks (ANN), activation functions such as ReLU, sigmoid, and tanh enable the network to learn complex patterns by modeling non-linear structures (H. A. N. Kubilay, G. Öztürk, & A. Aslan, 2023). The “ReLU Revolution,” which began with Nair and Hinton's introduction of ReLU (Rectified Linear Unit) in 2010, which enabled many applied CNN solutions in imaging and detection tasks (Aslan & Çelebi, 2022; Çelebi et al., 2024), revitalized deep learning and laid the foundation for today's successful models (Nair & Hinton, 2010).

Over the past decade, researchers have developed numerous variants to overcome ReLU's limitations: Leaky ReLU (Maas, Hannun, Ng, & others, 2013) PReLU (K. He et al., 2015), ELU (Clevert, Unterthiner, & Hochreiter, 2015) and SELU (Klambauer, Unterthiner, Mayr, & Hochreiter, 2017). Moreover, modern functions such as Swish (Ramachandran, Zoph, & Le, 2017),

discovered through automatic search methods, and GELU (Hendrycks, 2016) based on theoretical foundations, have become standards, especially in Transformer architectures. A comprehensive survey by (Kunc & Kléma, 2024) reveals that more than 400 activation functions have been proposed in the literature.

Classic Activation Functions

Sigmoid Function

Mathematical Definition

The sigmoid function is an S-shaped function that maps any real number to the interval (0,1), and its mathematical expression is given in Equation (2) (Vieira, 2020).

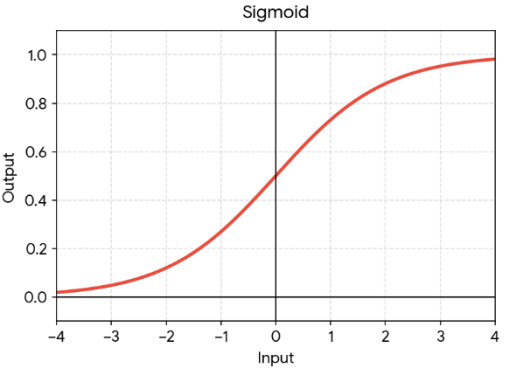
$$\underline{\underline{\sigma}}(x) = \frac{1}{(1 + e^{-x})} \quad (2)$$

The derivative has a form that can be defined in terms of itself, and this expression is shown in Equation (3).

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \quad (3)$$

This derivative reaches its maximum value of 0.25 at x = 0. This is a critical limitation because the gradient is multiplied by at most 0.25 in each layer during backpropagation. Figure 1 shows the sigmoid function.

Figure 1: Graphical representation of the sigmoid function



Advantages

- Smoothness: The function is differentiable at every point and infinitely differentiable (in class C∞). This property is ideal for optimization algorithms.
- Probability interpretation: Since the outputs lie in the interval (0, 1), they can be directly interpreted as probabilities in binary classification problems. This forms the basis of logistic regression.
- Efficient derivative calculation: The formula $\sigma'(x) = \sigma(x)(1-\sigma(x))$ does not require additional exponential processing in backpropagation because the value of $\sigma(x)$ in the forward pass is already calculated (Martinelli, Coraddu, & Cammarano, 2023).

Disadvantages

- Vanishing gradient problem: Because the maximum derivative is 0.25, the gradient in deep networks decays exponentially. In an n-layer network using sigmoid activation, the gradient contribution from activation derivatives alone can decay at approximately $(0.25)^n$. For 10 layers, this yields $0.25^{10} \approx 9.5 \times 10^{-7}$, severely limiting learning in early layers (Çelebi & Karaman, 2023).

- Not zero-centered: Outputs are in the range (0,1) with a mean around 0.5. This causes all inputs to neurons in the next layer to have the same sign, creating zig-zag dynamics in gradient updates.
- Saturation: For large absolute values of x($|x| \gg 1$), the derivative approaches zero. For large positive or negative inputs, neurons reach 'saturation' and learning stops (Han & Moraga, 1995; Yin, 2003).

Hyperbolic Tangent (Tanh)

Mathematical Definition

The hyperbolic tangent is a zero-centered version of the sigmoid function, mapping the outputs to the interval [-1,1] (Kumar & Sodhi, 2023). Its mathematical expression is given in Equation (4):

$$\tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \quad (4)$$

The tanh function can be rewritten in terms of the sigmoid function. This relationship is given in Equation (5).

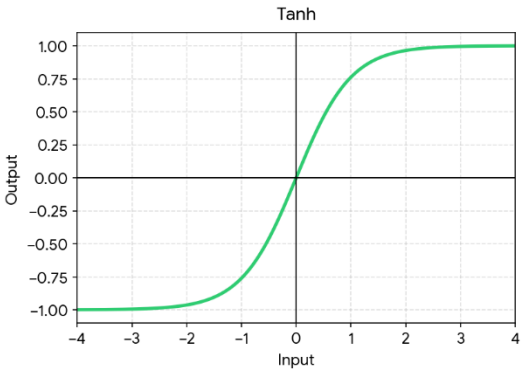
$$\tanh(x) = 2\sigma(2x) - 1 \quad (5)$$

The derivative of the tanh function can be expressed in terms of the function itself. This relationship is given in Equation (6).

$$\tanh'(x) = 1 - \tanh^{2}(x) \quad (6)$$

The maximum derivative is 1.0 at x=0, which provides a gradient 4 times stronger than the 0.25 of the sigmoid. Figure 2 shows a graph of the hyperbolic tangent (Ji, Feng, & Wang, 2025).

Figure 2: Hyperbolic tangent



Advantages

- Zero-centered output: The range [-1,1] and mean ≈0 are a significant improvement over sigmoid. This feature reduces problems with gradient updates.
- Stronger gradient: A maximum derivative of 1.0 provides four times more effective gradient propagation than sigmoid.
- Symmetric: Being symmetric about the origin (tanh(-x) = -tanh(x)) provides balanced learning for positive and negative inputs (Che-Wei Lin & Jeen-Shing Wang, 2008; Leboeuf, Namin, Muscedere, Wu, & Ahmadi, 2008).

Disadvantages

- Still a vanishing gradient: While better than the sigmoid, the derivative still approaches zero at large |x| values. The problem persists in deep networks.
- Computational cost: Requires two exponential calculations (ex and e-x) (Salah, Safar, Taher, & Salem, 2023).

Rectified Linear Unit (ReLU)

Mathematical Definition

ReLU is an extremely simple yet highly effective function. Its mathematical definition is given in Equation (7)(J. He, Li, Xu, & Zheng, 2018).

$$f(x) = \max(0, x) = \{ x, x \geq 0;\ 0,\ x < 0 \} \quad (7)$$

The derivative of the ReLU function is defined piecewise depending on the input value. This expression is given in Equation (8).

$$f'(x) = \{1, x > 0;\ 0,\ x < 0 \} \quad (8)$$

The derivative is undefined at x=0, but in practice it is usually taken to be 0 or 1.

Advantages

- Vanishing gradient solution: In the positive region, the derivative is 1.0, and there is no gradient decay. This feature made it possible to train deep networks.
- Computational efficiency: It is simply a thresholding operation (if x > 0 then x else 0), requiring no exponential computation. It runs extremely fast on GPUs.
- Sparsity: A significant portion of neurons can produce zero output, creating sparse representations. The exact sparsity depends on input distribution and network initialization.
- Fast convergence: (Krizhevsky et al., 2012) showed that training with ReLU converges approximately 6 times faster than with tanh.

Disadvantages: Dying ReLU Problem

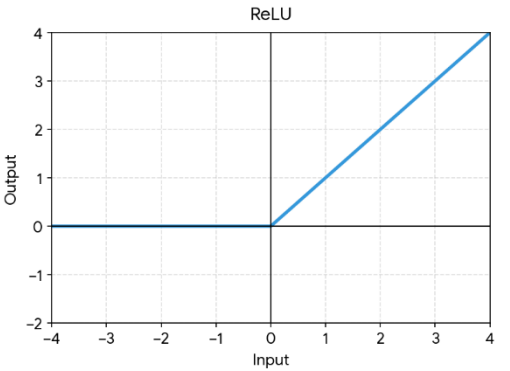
- The most serious problem with ReLU is the 'dying ReLU' problem. When a neuron's input is consistently negative, the output becomes 0 and the gradient also becomes 0, causing the neuron to become permanently inactive. Mathematically: If

$\sum(w_i x_i + b) < 0$for all inputs x, then, as shown in Equation (9)(Lu, Shin, Su, & Karniadakis, 2019):

$$\underline{f}(\sum w_i x_i + b) = \underline{\underline{0 \; and}} \; \frac{\partial f}{\partial w_i} = 0 \qquad (9)$$

In this case, the weights are never updated and the neuron dies. This problem can be particularly prevalent at high learning rates or poor initializations. Maas et al. (2013) observed that in some networks, 40% of neurons were inactive due to ReLU dying. Figure 3 shows the ReLU graph.

Figure 3: ReLU Graph



Vanishing Gradient Problem

The vanishing gradient problem is one of the most critical challenges in training deep neural networks (Hochreiter, 1998).

Mathematical Analysis

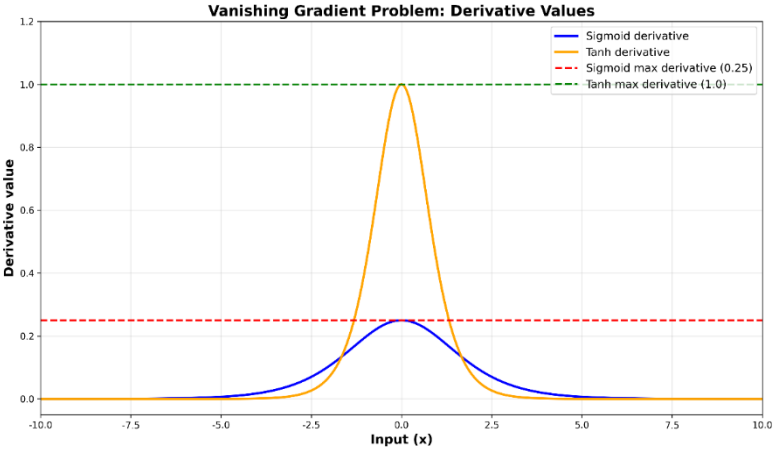
During backpropagation, the gradient for the weight $w\ i$ w i in the i-th layer is computed using the chain rule. This is given in Equation (10).

$$\partial L/\partial w_1 = \partial L/\partial a_n \cdot \partial a_n/\partial a_{n_1} \cdot \ldots \cdot \partial a_2/\partial a_1 \cdot \partial a_1/\partial w_1 \qquad (10)$$

Each term $\partial a_k/\partial a_{k_{-1}}$ includes the activation derivative, multiplying the gradient as it propagates backward through layers.

For sigmoid: Since $\sigma'(x) \leq 0.25$, the gradient decreases by approximately $(0.25)^n$ for n layers. 10 Layer: $0.25^{10} \approx 9.5 \times 10^{-7}$. This effectively means zero gradient, causing the initial layers to fail to learn. Figure 4 illustrates the vanishing gradient problem.

Figure 4: Vanishing gradient problem



Solution Approaches

Several techniques have been developed to mitigate the vanishing gradient problem:

- ReLU and its variants: These functions reduce gradient vanishing because their derivative is 1 in the positive region, preventing exponential decay.
- Weight initialization: Initialization methods ensure balanced gradient flow by scaling initial weights according to layer dimensions.
- Batch normalization: This technique normalizes activations between layers, stabilizing training and reducing sensitivity to initialization.

- Residual connections: ResNet architectures allow gradients to flow directly through skip connections, facilitating learning in very deep models (Agarap, 2018).

Modern ReLU Variants

Due to the dying ReLU problem of ReLU, researchers have developed several variants that aim to solve this problem.

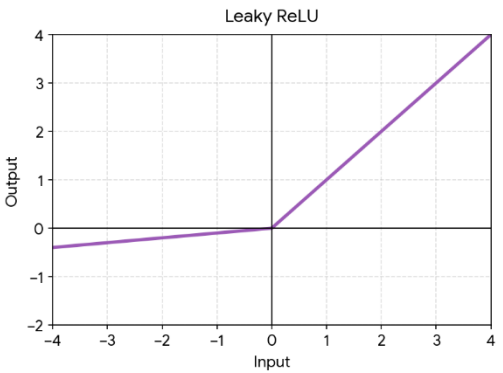
Leaky ReLU

Leaky ReLU addresses the dying ReLU problem by providing a small gradient in the negative region. Its mathematical definition is given in Equation (11)(Mastromichalakis, 2020).

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{11}$$

Here, α is typically 0.01. It was introduced for acoustic modeling by Maas, Hannun, and Ng (2013). The Leaky ReLu plot is given in Figure 5.

Figure 5: Leaky ReLU Graph



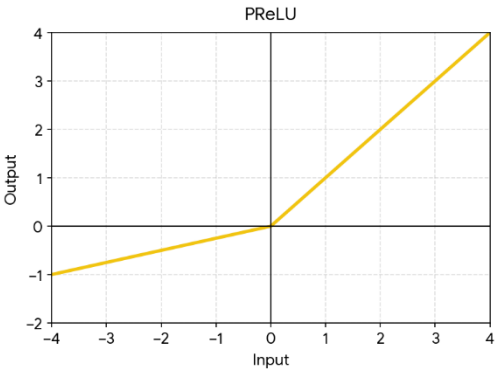
Parametric ReLU (PReLU)

PReLU generalizes Leaky ReLU by making the parameter αlearnable (Wang, Muhammad, Hong, Sangaiah, & Zhang, 2020). Its mathematical definition is given in Equation (12).

$$f(x) = \max(\alpha_i x, x) \quad (12)$$

(J. He et al., 2018) introduced PReLU and proposed the He initialization method, achieving a top-5 error rate of 4.94% on ImageNet, which exceeded human-level performance (5.1% error). The PReLU graph is shown in Figure 6.

Figure 6: PReLU Graph



Exponential Linear Unit (ELU)

ELU uses an exponential function in the negative region. Its mathematical definition is given in Equation (13).

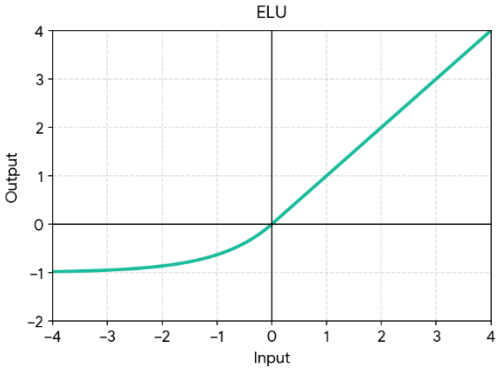
$$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases} \quad (13)$$

It was proposed by (Clevert et al., 2015). The ELU graph is given in Figure 7.

Advantages:

- ELU offers several advantages. It brings average activations closer to zero, producing a batch normalization-like effect. The function is smooth at x = 0 (C^1 continuous), and Clevert et al. (2015) demonstrated that it improves generalization in deep networks.

Figure 7: ELU Graph



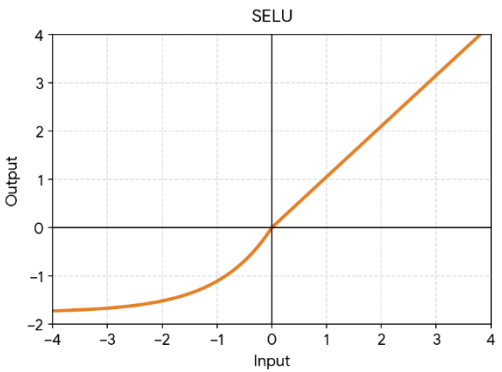
Scaled ELU (SELU)

SELU is an activation function specifically designed for self-normalizing networks. Its mathematical definition is given in Equation (14).

$$\text{selu}(x) = \lambda \cdot \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases} \tag{14}$$

Special parameters: $\alpha \approx 1.6733$, $\lambda \approx 1.0507$ (Klambauer et al., 2017) proved, using the Banach fixed point theorem, that SELU keeps the mean and variance of activations close to (0,1) across layers. This structurally prevents vanishing and exploding gradients. The SELU graph is shown in Figure 8.

Figure 8: SELU Graph



Advanced Activation Functions

Swish

Swish is an activation function discovered by Google Brain through automatic search methods (Mercioni & Holban, 2020). Its mathematical definition is given in Equation (15).

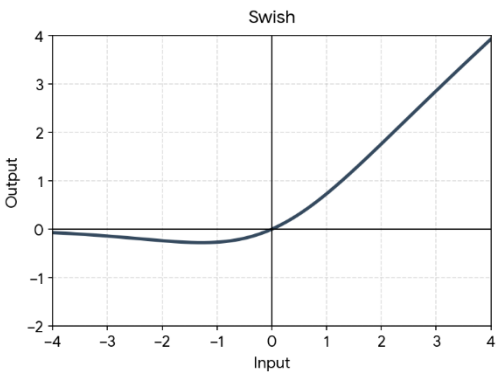
$$f(x)=x\cdot\sigma(\beta x) \quad (15)$$

(Ramachandran et al., 2017) tested 400+ functions in exhaustive search and found that Swish consistently outperforms ReLU. It is also called SiLU (Sigmoid Linear Unit) for $\beta = 1$. Figure 9 shows the Swish graph.

Features:

Swish exhibits several notable properties: it is smooth and infinitely differentiable, and unlike most popular activation functions, it is non-monotonic. Ramachandran et al. (2017) reported that Swish achieved a 0.9% improvement in top-1 accuracy on Mobile NASNet-A7 compared to ReLU.

Figure 9: Swish Chart



Gaussian Error Linear Unit (GELU)

GELU is a modern activation function designed based on probabilistic principles. Its mathematical definition is given in Equation (16).

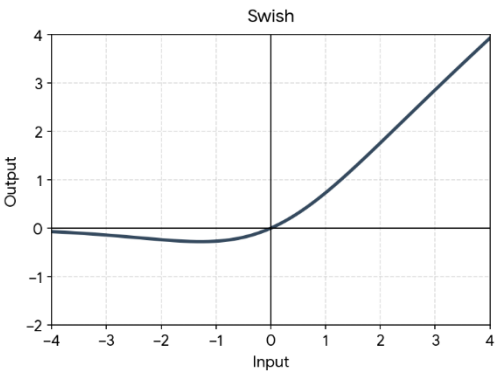
$$\mathrm{GELU}(x) = x \cdot \Phi(x) \tag{16}$$

Here, $\Phi(x)$ is the standard Gaussian cumulative distribution function. It was proposed by (Hendrycks, 2016). The GELU plot is shown in Figure 10.

Importance of GELU

It has become the standard activation in all major Transformer models, such as BERT, GPT-2, GPT-3, and Vision Transformer (ViT).

Figure 10: GELU Graph



Mish

Mish is a new activation function designed with self-regularizing features (Guo et al., 2025). Its mathematical definition is given in Equation (17).

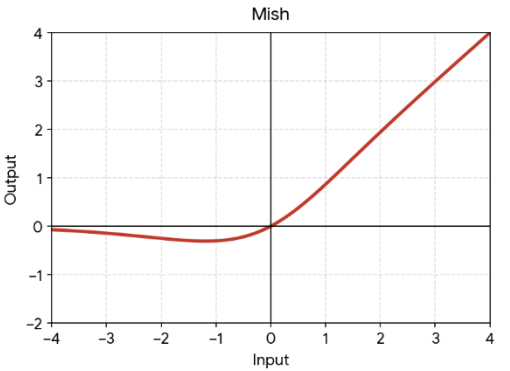
$$f(x) = x \cdot \tanh(\mathrm{softplus}(x)) = x \cdot \tanh(\ln(1 + e^{x})) \tag{17}$$

In YOLOv4, MS-COCO achieved a 2.1% AP improvement over Leaky ReLU. The Mish graph is shown in Figure 11.

Features:

Mish is smooth (C^{∞} continuous), exhibits non-monotonic self-gating behavior, and has an output range of approximately $[-0.31, \infty)$.

Figure 11: Mish Chart



Comparative Analysis and Implementation Suggestions

Summary Comparison

A general comparison of activation functions in terms of range, derivative properties, smoothness, monotonicity, and typical application areas is presented in Table 1. This table provides a quick overview of the strengths and limitations of each function.

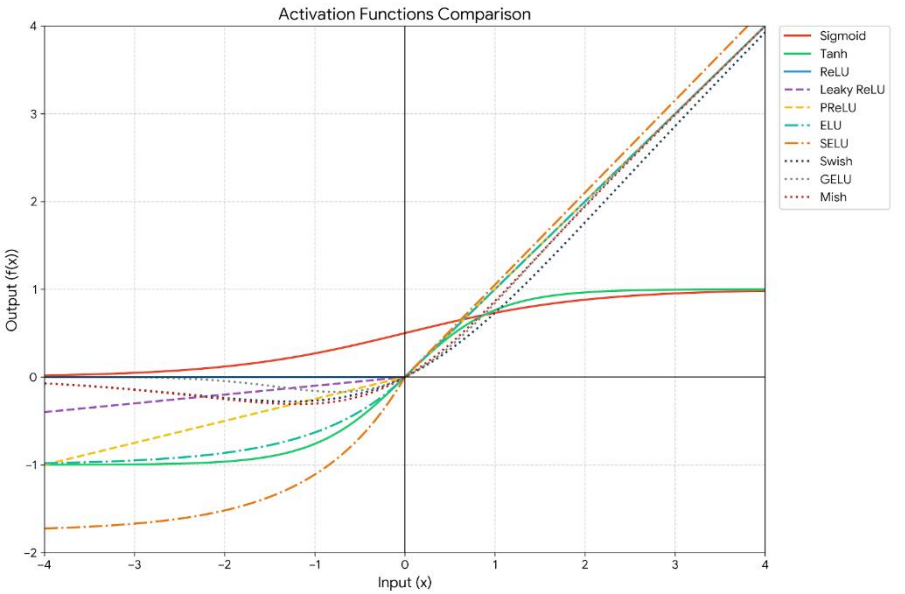
Table 1: Summary comparison of activation functions

Function	Range	Max Derivative	Smooth	Monotone	Main Use
Sigmoid	(0, 1)	0.25	✓	✓	Output (binary)
Tanh	[-1, 1]	1.0	✓	✓	RNN/LSTM
ReLU	[0, ∞)	1.0	✗	✓	Default
ELU	[-α, ∞)	1.0	✓	✓	Deep networks
Swish	[≈-0.28, ∞)	-	✓	✗	CNN

GELU	[≈-0.17, ∞)	-	✓	✗	Transformers
Mish	[≈-0.31, ∞)	-	✓	✗	YOLOv4

In Figure 12, the responses of commonly used activation functions—Sigmoid, Tanh, ReLU, Leaky ReLU, PReLU, ELU, SELU, Swish, GELU, and Mish—are presented in a comparative manner. The plot highlights differences in smoothness, output range, and nonlinear characteristics, providing a visual understanding of which activation functions may be more suitable for different deep learning architectures.

Figure 12: Comparison of all functions



Application Suggestions

Based on extensive empirical evidence, the following guidelines can help practitioners select appropriate activation functions:

- General deep networks: ReLU remains the default choice due to its computational efficiency, strong gradient propagation, and extensive empirical validation across diverse tasks.
- Very deep networks (>50 layers): Swish or Mish provide smoother gradient flow and reduce vanishing gradient effects, particularly in networks with residual connections.
- Transformer architectures: GELU has become the standard in models like BERT, GPT, and Vision Transformer (ViT), offering theoretical grounding in stochastic regularization.
- Object detection: Mish has demonstrated superior performance in YOLOv4, achieving a 2.1% AP improvement over Leaky ReLU on MS-COCO.
- Binary classification output layer: Sigmoid enables direct probability interpretation for binary decisions.
- Multi-class output layer: Softmax produces a valid probability distribution over classes.

Conclusion

Activation functions are fundamental building blocks that enable neural networks to model nonlinear relationships. The field has evolved from the early dominance of sigmoid and tanh through the ReLU revolution, culminating in modern smooth functions like GELU and Mish that balance theoretical principles with empirical performance. While ReLU remains the default choice for general architectures due to its simplicity and effectiveness, specialized functions have emerged for specific domains. GELU has become standard in Transformer models, while Mish shows promise in object detection tasks. Despite more than 400 proposed activation functions, only a handful have achieved widespread adoption, suggesting that practical performance and computational efficiency matter more than theoretical novelty. The selection of activation

functions significantly impacts model performance, yet no single function is universally optimal. Practitioners must consider task requirements, network depth, computational constraints, and empirical evidence when choosing activations. Future research may explore adaptive activation functions that adjust their behavior based on learned representations, or architecture-specific designs that exploit domain knowledge. As deep learning continues to advance, activation functions will remain a critical area where mathematical insight and empirical experimentation drive progress.

References

Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU) (Version 2). Version 2. arXiv. https://doi.org/10.48550/ARXIV.1803.08375

Aslan, A. (2025). Deep Learning in Neurological Imaging: A Novel CNN-Based Model for Brain Tumor Classification Türkiye And Health Risk Assessment. İnönü Üniversitesi Sağlık Hizmetleri Meslek Yüksek Okulu Dergisi, 13(2), 457–474. https://doi.org/10.33715/inonusaglik.1645318

Aslan, A., & Çelebi, S. B. (2022). Real Time Deep Learning Based Age and Gender Detection For Advertising and Marketing. Uluslararası Bilişim Kongresi (IIC 2022): Bildiriler Kitabı, 10–16.

Birecikli, B., Karaman, Ö. A., Çelebi, S. B., & Turgut, A. (2020). Failure load prediction of adhesively bonded GFRP composite joints using artificial neural networks. Journal of Mechanical Science and Technology, 34(11), 4631–4640. https://doi.org/10.1007/s12206-020-1021-7

Çalişkan, A. (2022). Classification of Tympanic Membrane Images based on VGG16 Model. Kocaeli Journal of Science and Engineering, 5(1), 105–111. https://doi.org/10.34088/kojose.1081402

Çelebi, S. B., Aslan, A., & Canpolat, M. (2026). Predicting Gelation in Copolymers Using Deep Learning Through a Comparative Study of ANN, CNN, and LSTM Models with SHAP Explainability. In G. Bergami, P. Ezhilchelvan, Y. Manolopoulos, S. Ilarri, J. Bernardino, C. K. Leung, & P. Z. Revesz (Eds.), Database Engineered Applications (pp. 85–95). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-06744-9_7

Çelebi, S. B., & Karaman, Ö. A. (2023). Multilayer LSTM Model for Wind Power Estimation in the Scada System. European Journal of Technic. https://doi.org/10.36222/ejt.1382837

Çelebi, S. B., Karaman, Ö. A., & Öztürk, G. (2024). Autonomous Driving Technologies and Cyber Security in Vehicles. In Bilgisayar Bilimleri ve Mühendisliğinde İleri Araştırmalar. Yaz Yayınları.

Che-Wei Lin & Jeen-Shing Wang. (2008). A digital circuit design of hyperbolic tangent sigmoid function for neural networks. 2008 IEEE International Symposium on Circuits and Systems (ISCAS), 856–859. Seattle, WA, USA: IEEE. https://doi.org/10.1109/ISCAS.2008.4541553

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv Preprint arXiv:1511.07289, 4(5), 11.

Deng, L., & Liu, Y. (2018). Deep learning in natural language processing. Springer.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139–144. https://doi.org/10.1145/3422622

Guo, Q., Zhao, M., Liu, Y., Wu, B., Yang, D., Qiao, M., … Xie, W. (2025). Developing an attention-enhanced deep learning approach for impurity detection in Camellia oleifera seeds. Journal of Food Composition and Analysis, 148, 108148. https://doi.org/10.1016/j.jfca.2025.108148

H. A. N. Kubilay, G. Öztürk, & A. Aslan. (2023). Yapay Sinir Ağları Kullanarak Yüzey Pürüzlülüğü Tespiti. 1, 487–492.

Han, J., & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In J. Mira & F. Sandoval (Eds.), From Natural to Artificial Neural Computation (pp. 195–201). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-59497-3_175

He, J., Li, L., Xu, J., & Zheng, C. (2018). ReLU Deep Neural Networks and Linear Finite Elements. https://doi.org/10.48550/ARXIV.1807.03973

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE International Conference on Computer Vision, 1026–1034.

Hendrycks, D. (2016). Gaussian error linear units (gelus). arXiv Preprint arXiv:1606.08415.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 06(02), 107–116. https://doi.org/10.1142/S0218488598000094

Ji, X., Feng, M., & Wang, J. (2025). Characterization of the thermocline in the eastern Pacific Ocean based on a hyperbolic tangent function fitting method. Ocean Dynamics, 75(9), 81. https://doi.org/10.1007/s10236-025-01726-y

Karaman, Ö. A. (2023). Prediction of Wind Power with Machine Learning Models. Applied Sciences, 13(20), 11455. https://doi.org/10.3390/app132011455

Karaman, Ö. A., Çelebi, S. B., & Öztürk, G. (2024). Electric Vehicle Charging Stations and Harmonic Problem. In Research and Evaluations in the Field of Electrical-Electronics and Communication Engineering. Gece Kitaplığı.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. Advances in Neural Information Processing Systems, 30.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.

Kumar, A., & Sodhi, S. S. (2023). Some Modified Activation Functions of Hyperbolic Tangent (TanH) Activation Function for Artificial Neural Networks. In A. Bhattacharya, S. Dutta, P. Dutta, & V. Piuri (Eds.), Innovations in Data Analytics (pp. 369–392). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-99-0550-8_30

Kunc, V., & Kléma, J. (2024). Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks (Version 1). Version 1. arXiv. https://doi.org/10.48550/ARXIV.2402.09092

Leboeuf, K., Namin, A. H., Muscedere, R., Wu, H., & Ahmadi, M. (2008). High Speed VLSI Implementation of the Hyperbolic Tangent Sigmoid Function. 2008 Third International Conference on Convergence and Hybrid Information Technology, 1070–1073. Busan, Korea: IEEE. https://doi.org/10.1109/ICCIT.2008.131

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., & Aulagnier, S. (1996). Application of neural networks to modelling nonlinear

relationships in ecology. Ecological Modelling, 90(1), 39–52. https://doi.org/10.1016/0304-3800(95)00142-5

Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2019). Dying ReLU and Initialization: Theory and Numerical Examples. https://doi.org/10.48550/ARXIV.1903.06733

Maas, A. L., Hannun, A. Y., Ng, A. Y., & others. (2013). Rectifier nonlinearities improve neural network acoustic models. Proc. Icml, 30, 3. Atlanta, GA.

Martinelli, C., Coraddu, A., & Cammarano, A. (2023). Approximating piecewise nonlinearities in dynamic systems with sigmoid functions: Advantages and limitations. Nonlinear Dynamics, 111(9), 8545–8569. https://doi.org/10.1007/s11071-023-08293-1

Mastromichalakis, S. (2020). ALReLU: A different approach on Leaky ReLU activation function to improve Neural Networks Performance (Version 2). Version 2. arXiv. https://doi.org/10.48550/ARXIV.2012.07564

Mercioni, M. A., & Holban, S. (2020). P-Swish: Activation Function with Learnable Parameters Based on Swish Activation Function in Deep Learning. 2020 International Symposium on Electronics and Telecommunications (ISETC), 1–4. Timisoara, Romania: IEEE. https://doi.org/10.1109/ISETC50328.2020.9301059

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10), 807–814.

Nosouhian, S., Nosouhian, F., & Khoshouei, A. K. (2021). A review of recurrent neural network architecture for sequence learning: Comparison between LSTM and GRU.

Öztürk, G., Aslan, A., & Fırat, M. (2025). The effect of dibutyl maleate additive on combustion characteristics in compression ignition engines and analysis of performance prediction based on machine learning.

Öztürk, G., Çelebi, S. B., & Karaman, Ö. A. (2024). Autonomous Vehicles: Systems, Technoloies and Sensors. In Otomotiv Mühendisliğinde Yenilikçi Teknolojiler, Performans Çözümleri ve Geleceğin Sistemleri (1st ed., pp. 34–81). Bidge Yayınları. https://doi.org/10.70269/10.70269/6166236563

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for Activation Functions (Version 2). Version 2. arXiv. https://doi.org/10.48550/ARXIV.1710.05941

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

Salah, K., Safar, M., Taher, M., & Salem, A. (2023). Configurable model for sigmoid and hyperbolic tangent functions. Applications of Modelling and Simulation, 7, 78–84.

Sejnowski, T. J. (2018). The deep learning revolution. MIT press.

Sunar, A. S., & Khalid, M. S. (2024). Natural Language Processing of Student's Feedback to Instructors: A Systematic Review. IEEE Transactions on Learning Technologies, 17, 741–753. https://doi.org/10.1109/TLT.2023.3330531

Tekin, T. G. (2023). Geniş ekonomik gruplarin ihracat değerinin yapay sinir ağlari yöntemiyle tahminlenmesi. EKEV Akademi Dergisi, (95), 319–335.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vieira, V. (2020). Solutions for better fitting Sigmoid-shaped functions to binary data. Computational Ecology and Software, 10(4), 162.

Wang, S.-H., Muhammad, K., Hong, J., Sangaiah, A. K., & Zhang, Y.-D. (2020). RETRACTED ARTICLE: Alcoholism identification via convolutional neural network based on parametric ReLU, dropout, and batch normalization. Neural Computing and Applications, 32(3), 665–680. https://doi.org/10.1007/s00521-018-3924-0

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Wythoff, B. J. (1993). Backpropagation neural networks. Chemometrics and Intelligent Laboratory Systems, 18(2), 115–155. https://doi.org/10.1016/0169-7439(93)80052-J

Yin, X. (2003). A Flexible Sigmoid Function of Determinate Growth. Annals of Botany, 91(3), 361–371. https://doi.org/10.1093/aob/mcg029

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation, 31(7), 1235–1270. https://doi.org/10.1162/neco_a_01199

BÖLÜM 5

A SURVEY OF WORD EMBEDDING

Ümmügülsüm MENGUTAYCI[1]

Introduction

The increasingly digital age has also brought about an increase in text data on the internet.Online news sites, e-commerce applications, databases providing access to academic publications, social media applications, and the data of institutions and organizations are all composed of text. For example, automatically determining which category a news item to be added to a news site belongs to eliminates a workload.For example, automatically determining which category a news article to be added to a news site belongs toeliminates a workload. On online shopping sites, the analysis of customer reviews is important in terms of giving ideas to new customers who will make a purchase.In order to perform a semantic analysis of all this data, word representations must be made on text data.This article focuses on word representation methods frequently used in the literature and explains the working principles of these systems.Word representation, also known as word embedding, aims to capture the semantic and linguistic relationships between words (Wang et al., 2018). Word representations are used in sentiment analysis (Altowayan & Tao, 2016; Deho, Agangiba, Aryeh, & Ansah, 2018;

[1] Res.Asst., Tarsus University, Department of Computer Engineering, Orcid: 0000-0001-9861-8957

Giatsoglou et al., 2017; Yu, Wang, Lai, & Zhang, 2017), entity recognition systems (Naseem, Musial, Eklund, & Prasad, 2020; Park et al., 2015; Segura-Bedmar, Suárez-Paniagua, & Martínez, 2015), medical applications (Naseem et al., 2020; Shao, Taylor, Marshall, Morioka, & Zeng-Treitler, 2019; Wang et al., 2018), question-answering systems (Medved' & Horák, 2018; Othman, Faiz, & Smaïli, 2019) and translation systems (Tian, Song, Ting, & Huang, 2022). This study presents examples of recent work in different fields using word representation methods. This study provides a comparative overview of popular word representation methods.

Term Frequency-Inverse Document Frequency (TF-IDF)

It attempts to determine how important the selected word is in the relevant corpus (Ramos, 2003). The TF value calculates the frequency of the relevant term within a given document. The IDF value expresses the ratio of the corpus size to the frequency of the selected term occurring in the entire corpus. The TF-IDF value is obtained by multiplying the TF value by the logarithm of the IDF value, as shown in Equation 1 (Trstenjak, Mikac, & Donko, 2014).

$$t_{ij} = tf_{ij} idf_i = tf_{ij} \times \log_2(N \div df_i) \quad (1)$$

Unless specifically searched for within a collection, pronouns, prepositions, and conjunctions are considered insignificant within a document, and the TF-IDF value of these words is very low. Thus, these words can be removed from the text by setting a specific threshold value. The TF-IDF word representation model, despite its efficient operation and simple algorithmic structure, may pose problems in very large datasets because it does not examine the relationships between words (Ramos, 2003).

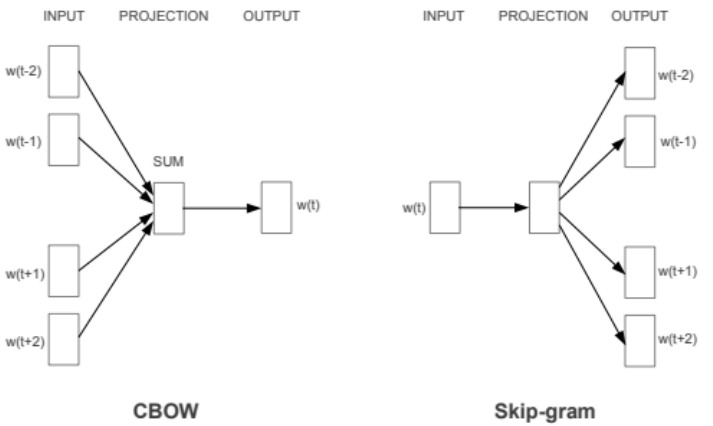
Word2vec

It is a word representation method proposed by Mikolov and colleagues. It is based on two architectures:

- ***Continuous Bag of Words(CBOW):*** The average of all word vectors is taken, and words are shared in the same position. The desired word is predicted by taking the preceding and following words as input. The order of the words is not important.
- ***Continuous Skip-Gram:*** Unlike the CBOW model, the current word at the desired position is taken as input and the words before and after it are predicted (Mikolov, Chen, Corrado, & Dean, 2013).

The working principle of the CBOW and Skip-gram models is illustrated in Figure 1.

Figure 1. CBOW and Skip-gram working structure (Mikolov et al., 2013)



FastText

It is a word representation model library created by the Facebook research center in 2016. Approaches such as Word2Vec and TF-IDF create a vector representation for each word. The roots of words are not examined. This situation may impose some limitations in morphologically rich languages such as Turkish and Finnish. For example, in Turkish, a verb can take more than one inflectional suffix. The fact that a word can have more than one form can affect training success. To overcome this problem, researchers have divided words into n-grams. Thus, they have managed to represent word roots. Each word is represented by the sum of its n-grams. In the Fasttext representation model, each word is divided into subwords, requiring no preprocessing and enabling fast training of the dataset (Bojanowski, Grave, Joulin, & Mikolov, 2017).

Global Vectors for Word Representation (GLOVE)

It is an unsupervised learning algorithm. It calculates the ratios of the probabilities of words appearing together within a corpus. For example, the word "gender" can be associated with "woman","man", "brother" and "sister" and their probabilities are very close to each other, to the extent that they are equal.

The Glove model uses weighted least squares regression. It calculates cosine similarity to measure the relationships between words (Pennington, Socher, & Manning, 2014). Cosine similarity represents the cosine value of the angle between two word vectors. It roughly attempts to find the similarity measure between two vectors. The cosine similarity of the two word vectors a and b given in Equation 2 is formulated. $\|a\|$, is the result of the Euclidean relation of the given vector (Han, Kamber, & Pei, 2012).

$$sim(a,b) = \frac{a \times b}{\|a\|\|b\|} \qquad (2)$$

A corpus is divided into semantic (based on person, country, gender) and syntactic (verb, adjective) subsets. Semantically, it finds the answer to the question: *If Ankara is to Turkey*, then what is *Paris to*

...? Syntactically, in a corpus where the words water-watering are related, the word sick-become sick can be associated with them. As a result, it has been emphasized that the Glove model performs better than other models (Word2Vec) in word analogy, word similarity, and named entity recognition tasks (Pennington et al., 2014).

Transformers

A transformer has a six-layer encoder and a six-layer decoder structure. The encoder layer has two sub-layers: multi-head self attention and a position-based feedforward network layer. The decoder layer has a third sub-layer that performs masked multi-head attention, in addition to the two sub-layers in the encoder layer.

Attention is the mapping of a key-value pair, where the keys, values and all outputs are vectors, to an output. The multi-head attention mechanism enables the processing of multi-dimensional keys, values and queries to produce multi-dimensional outputs.

Transformers use the multi-head attention operation in three different ways;

- Encoder-decoder attention layer: Queries for this layer come from the previous decoder layer. Memory keys and values are taken from the encoder output. Each position in the decoder is connected to all positions in the input sequence.
- In the second case, the encoder incorporates a self-attention mechanism. Thus, all positions in the encoder are connected to all positions in the previous layer, and all queries, values, and keys are taken from the output of the previous layer in the encoder.
- Finally, there may be self-attention in the decoder. In this case, each position in the decoder is connected to other positions (Vaswani et al., 2017).

OpenAI- Generative Pre-trained Transformer (GPT)

It is a word embedding model that uses a combination of unsupervised pre-training and supervised fine-tuning. It uses a unidirectional language model. This word embedding method uses a multi-layer Transformer decoder. This model produces output for the target token by applying multi-head self-attention on input tokens and position-based feedforward layers (Radfort, Narasimhan, Salimans, & Sutskever, 2018).

Embedding from Language Models (ELMO)

This model is a deep contextualized word representation method that models the complex features of word usage, such as semantic and syntactic properties and how this usage changes across linguistic contexts. In this model, unlike other methods, each token is assigned a representation vector that is a function of the input sentence. Thus, the same word can have different word vectors in different contexts. It uses vectors derived from a bidirectional Long Short Term Memory (LSTM) model. Word vectors are computed on a pre-trained bidirectional language model (biLM) (Peters et al., 2018).

Bidirectional Encoder Representations from Transformers (BERT)

Input representations encompass token sequences consisting of both single sentences and sentence pairs. The first token of each input sequence is a special classification token [CLS]. Sentence pairs are grouped together in a single sequence for classification. Sentences are separated in two different ways: either by the [SEP] token or by adding a representation to each token indicating whether it belongs to sentence A or B. The input value of each token is found by summing the position, segment (which sentence it belongs to), and the corresponding token value.

It is a pre-trained language model. It is used to predict the relationship between sentences such as entity recognition and question-answering. It has two approaches:

- ***Feature-based:*** It uses a left-to-right language model. It is commonly used for sentence embedding and paragraph embedding.
- ***Fine-tuned:*** It works to find pre-trained word embedding parameters from unlabeled data, similar to feature-based methods. The advantage of this approach is that only a small number of parameters need to be learned from scratch.

Both applications use a unidirectional architecture during pre-training to learn language representations. They use a left-to-right language model. That is, each token is concatenated to the previous token. This limitation falls short in sentence-level tasks such as question-answering. It is more important to include both directions in training, both right-to-left and left-to-right, for each token. That is why a fine-tuned BERT model is recommended. It uses the Masked Language Model (MLM) and NSP (Next Sentence Prediction) steps during the pre-training phase. When a sentence is input into the model, 15% of the words in the sentence are masked using the MLM technique. The purpose of masking is to predict the original word based solely on its context. In the MLM technique, the masked word is predicted using the explicitly fed words. Thus, instead of reconstructing all input values, the masked words are predicted. Although the MLM method enables the creation of a bidirectional pre-trained model, it creates inconsistency between the pre-training and fine-tuning steps because [MASK] tokens are not visible during fine-tuning. To reduce this inconsistency, masked words are not always replaced with the [MASK] token:

- 80% of the selected 15% of words are replaced with the [MASK] token.
- 10% are replaced with a random word.
- The remaining 10% are not replaced.

NSP is important for question-answering and natural language representation. In NSP, relationships between sentences are established. During training, for each pair of sentences that come in tandem, it is predicted whether the second sentence is a continuation

of the first sentence. For each pre-trained example, in the selected sentences A and B, sentence B is either 50% of the time the sentence following A (labeled as IsNext) or 50% of the time a random sentence within the corpus (labeled as NotNext) (Devlin, Chang, Lee, Google, & Language, 2018).

Over time, derivatives of the BERT model have been developed, such as ALBERT, BART, and RoBERTa (Topal, Bas, & Van Heerden, 2021).

Generalized Autoregressive Pretraining for Language Understanding (XLNET)

In the BERT model, some data is replaced with the [MASK] token during the pre-training step, but this token is not present in the actual data during the fine-tuning phase. This causes inconsistency between pre-training and fine-tuning. Furthermore, since the predicted tokens in the input are masked, they are considered independent of each other when considering the unmasked data. XLNET was developed to address these shortcomings in the BERT model.

XLNET uses autoregressive language model (AR) and autoencoder (AE) unsupervised pre-training models during the pre-training step. The AR model expresses the probability distribution of texts in a corpus together with the autoregressive model. The AE model reconstructs the original data from corrupted input.

The XLNET model maximizes all possible permutation probabilities of a sequence. This allows the context of each position to consist of tokens from both the right and left sides. It has a bidirectional context modeling capability (Yang et al., 2019).

References

Altowayan, A. A., & Tao, L. (2016). Word embeddings for Arabic sentiment analysis. *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, 3820–3825. https://doi.org/10.1109/BIGDATA.2016.7841054

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching Word Vectors with Subword Information*. 135–146. https://doi.org/10.1162/tacl_a_00051/1567442/tacl_a_00051.pdf

Deho, O. B., Agangiba, W. A., Aryeh, F. L., & Ansah, J. A. (2018). Sentiment analysis with word embedding. *IEEE International Conference on Adaptive Science and Technology, ICAST*, *2018-August*. https://doi.org/10.1109/ICASTECH.2018.8506717

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Retrieved from https://github.com/tensorflow/tensor2tensor

Giatsoglou, M., Vozalis, M. G., Diamantaras, K., Vakali, A., Sarigiannidis, G., & Chatzisavvas, K. C. (2017). Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, *69*, 214–224. https://doi.org/10.1016/J.ESWA.2016.10.043

Han, J., Kamber, M., & Pei, J. (2012). Getting to Know Your Data. *Data Mining*, 39–82. https://doi.org/10.1016/B978-0-12-381479-1.00002-2

Medved', M., & Horák, A. (2018). *Sentence and Word Embedding Employed in Open Question-Answering*. https://doi.org/10.5220/0006595904860492

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International*

Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings.

Naseem, U., Musial, K., Eklund, P., & Prasad, M. (2020). Biomedical Named-Entity Recognition by Hierarchically Fusing BioBERT Representations and Deep Contextual-Level Word-Embedding. *Proceedings of the International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN48605.2020.9206808

Othman, N., Faiz, R., & Smaïli, K. (2019). Enhancing Question Retrieval in Community Question Answering Using Word Embeddings. *Procedia Computer Science*, *159*, 485–494. https://doi.org/10.1016/J.PROCS.2019.09.203

Park, C. Y., Kim, Y.-S., Seok, M., Song, H.-J., Park, C.-Y., & Kim, J.-D. (2015). *Comparison of NER Performance Using Word Embedding*. https://doi.org/10.14257/astl.2015.120.154

Pennington, J., Socher, R., & Manning, C. D. (2014). *GloVe: Global Vectors for Word Representation*. 1532–1543. Retrieved from http://nlp.

Peters, M. E., Neumann, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations*. Retrieved from http://allennlp.org/elmo

Radfort, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). (OpenAI Transformer): Improving Language Understanding by Generative Pre-Training. *OpenAI*.

Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the First Instructional Conference on Machine Learning*.

Segura-Bedmar, I., Suárez-Paniagua, V., & Martínez, P. (2015). *Exploring Word Embedding for Drug Name Recognition*. 64–72. Retrieved from http://www.drugbank.ca

Shao, Y., Taylor, S., Marshall, N., Morioka, C., & Zeng-Treitler, Q. (2019). Clinical Text Classification with Word Embedding Features vs. Bag-of-Words Features. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2874–2878. https://doi.org/10.1109/BIGDATA.2018.8622345

Tian, T., Song, C., Ting, J., & Huang, H. (2022). A French-to-English Machine Translation Model Using Transformer Network. *Procedia Computer Science*, *199*, 1438–1443. https://doi.org/10.1016/J.PROCS.2022.01.182

Topal, M. O., Bas, A., & Van Heerden, I. (2021). *Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet*. Retrieved from https://towardsdatascience.com/Transformers-141e32e69591

Trstenjak, B., Mikac, S., & Donko, D. (2014). KNN with TF-IDF based Framework for Text Categorization. *Procedia Engineering*, *69*, 1356–1364. https://doi.org/10.1016/J.PROENG.2014.03.129

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., … Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems* (Vol. 30).

Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., … Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, *87*, 12–20. https://doi.org/10.1016/J.JBI.2018.09.008

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). *XLNet: Generalized Autoregressive Pretraining for*

Language Understanding. Retrieved from https://github.com/zihangdai/xlnet

Yu, L.-C., Wang, J., Lai, K. R., & Zhang, X. (2017). *Refining Word Embeddings for Sentiment Analysis*. 534–539.

BÖLÜM 6

THE EVOLUTION OF COVERT CHANNEL DETECTION: A COMPREHENSIVE REVIEW OF METHODS AND CHALLENGES

Şule YÜCELBAŞ[1]
Cüneyt YÜCELBAŞ[2]

INTRODUCTION

Covert channels in cyber security are a way for two people to communicate, in this case, using a computer, which do not have authorization to send or receive communications. Covert channels operate outside of standard security policy and access controls by leveraging file systems, memory structures, or protocols where security systems cannot directly monitor or block them hence allowing the passage of confidential information. Therefore, the role of covert channels plays an important role in the research of computer security and network security. The first investigations into covert channels began with defensive system security research in the 1970s, specifically due to application reports like Lampson (1973) and Bell and LaPadula (1976). During this time the primary concern

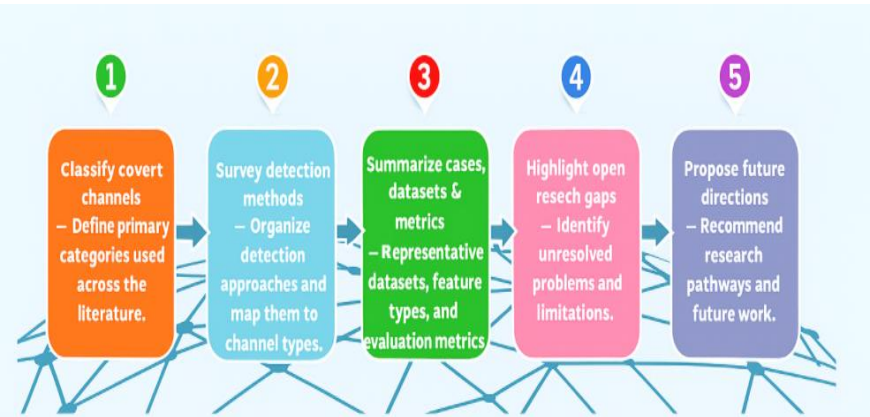
[1] Assoc.Prof.Dr., Tarsus University, Department of Computer Engineering, Mersin, Türkiye Orcid: 0000-0002-6758-8502
[2] Assoc.Prof.Dr., Tarsus University, Department of Electronics and Automation, Mersin, Türkiye Orcid: 0000-0002-6758-8502

was detecting when authorized system users violate the system or access control; however, in the 1990s and 2000s, the emphasis shifted to the growing popularity of network-based applications.

Since late 2015, covert channel research has gone through several major shifts in focus. The first major driving force during this time was the growth of secure methods of communicating over the internet using methods like TLS, DoH, DoT, and QUIC. This has allowed attackers to have much better chances of avoiding detection using traditional methods of identifying malicious activity by creating content within packets that cannot be identified or filtered using signature-based techniques. The second driving factor is that as cyber threats have become increasingly sophisticated, and as the number of APTs has increased, the use of low-bandwidth and hard-to-detect methods of transferring information from a system has made covert channels much more strategically important for attackers. Lastly, the maturity of deep learning and machine learning technologies has given attackers the ability to develop sophisticated covert channels, and defenders have developed much better ways to detect them by utilizing deep learning and machine learning to find patterns and extract information from a variety of different types of signals. Creating a path for the transition in quantity of the literature regarding covert channel research has progressed from using traditional statistical techniques to using hybrid machine learning methods such as Random Forests and LSTMs, and now to deep learning approaches, including CNNs and Transformers (according to Elsadig & Gafar, 2022; Caviglione, 2021). In addition, the production of many types of data sets and the consideration of a wide variety of use case scenarios (e.g., DNS-based DoH, TCP timing measurements, and how to modify packet size) for performance metric establishment has led to an expansion in literature about the topic and allowed for an evolution of the direction of that literature.

The technology-based focus of the last decade has not only been based upon technology, but rather the evolving needs of the cybersecurity environment also led to new areas of research and development in real-time monitoring, low latency detection, and generalizable defense techniques. The authors consider the timeframe of 2015-2025 as a significant period for investigating covert channels because of their diversity in both application and technological maturity. This article will systematically review the existing literature from 2015-2025 regarding the detection of covert channel communications in both the network and physical environments. A flowchart of the processes used to conduct the review has been included in Figure 1.

Figure 1: Procedural outline of this systematic review for the 2015–2025 period



This diagram outlines an organized process comprising five primary steps. First, classify covert channels into four primary categories to develop a foundational taxonomy. Second, conduct a survey of currently utilized techniques (statistical analysis, signature-based detection, machine learning, and deep learning) for detecting covert channels. Third, compile a summary of representative use cases, datasets, and evaluation metrics related to detecting covert channels to create an accumulation of empirical

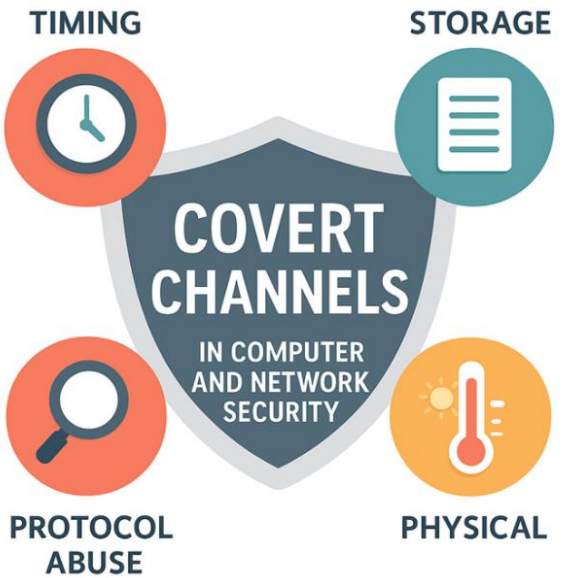
knowledge regarding how covert channel detection techniques function and the typical datasets used when developing those systems. Fourth, identify areas where research remains incomplete. Multiple open areas have been identified that include benchmarks and the lack thereof, limited generalizability to new data sets, adversarial implications, and latency challenges. Lastly, point out potential paths for future research that require domain adaptability, adversarial robustness, and low-latency solutions to enable continued research in this area.

HIDDEN CHANNEL CATEGORIES

In the literature, covert channels can be divided into four basic classifications, as detailed in Figure 2. Timing channels transmit information through purposely introduced changes to the transmission delay or time between packets. For example, the sender can send packets or hold one back for a time frame that coincides with an individual bit value. The variations from typical packet transmission create significant challenges with regard to utilizing signature-based methods for detection and analysis, especially in the case of encrypted data. In the case of Storage Channels, information is transmitted through changes made in fields that are not used in a protocol header, in packet length, or in other data structures. Storage Channels are primarily used to facilitate the transfer of information indirectly, for instance through TCP/IP, DNS, or HTTP headers. Protocol abuse channels are used inappropriately or in a way other than intended by the original developer of the protocol. For example, an individual could use DNS queries as a means to retrieve files, or use any changes made to the HTTP headers for similar purposes. The Technology Abuse Channel has been proposed to describe the use of traditional UDP packets to conceal the transmission of information over the Internet between parties who did not intend such usage. Lastly, Physical Channels transmit information in ways that go across established network boundaries and use methods of data

transmission that rely on other means, such as optical, acoustic, thermal, or electromagnetic (or "acoustic") devices. The use of a display screen's light intensity to indicate information is an example of a Physical Covert Channel (Ayub et al., 2019; Caviglione, 2021).

Figure 2: Categories of covert channels in computer and network security



The classification used in defence mechanisms and the areas they focus on, giving researchers a structured framework for analysing their results. There are recent examples of more detailed studies of sub-types of the four categories, along with other studies examining hybrid attack scenarios (Ayub et al., 2019; Caviglione, 2021). For example, the use of timing and storage channel combinations allows an attacker to evade detection using both in-network and physical surveillance. More recently in the literature, detection techniques for covert channel analysis have transitioned from traditional statistical analysis to ML and DL based methods. Recent findings have shown that these techniques are effective in

detecting emerging and complex patterns in network traffic from high-traffic volume and/or encrypted traffic. Each of the four classes will be discussed in detail, along with a review of the detection methods that fall under the categories of statistical analysis, signature-based detection, machine learning, deep learning, image representation, and non-protocol specific detection.

COMPREHENSIVE REVIEW OF DETECTION METHODS

Statistical and Rule-Based Methods

Statistical and rule-based methods constitute the earliest and most widely used class of covert channel detection. These approaches are based on detecting deviations from normative distributions of quantitative attributes of network traffic, such as time-domain properties (e.g., inter-arrival time, packet inter-departure time), size-domain properties (packet length distributions, payload length statistics), information content measures (entropy, Bayesian/complexity metrics), and frequency/spectral analysis. A typical workflow involves (i) computing appropriate statistical attributes, (ii) constructing benign traffic profiles, and (iii) comparing observed traffic with this profile. This class has the advantage of being amenable to real-time or near-real-time monitoring with low computational overhead and can be highly sensitive to known types of attacks. However, they also have distinct disadvantages: low-bandwidth, adaptive, or hybrid (timing + storage) channels; They can exhibit high false-negative rates against attackers who mask statistical signatures by adding random delays/schemes and modulations that mimic normal variations of the protocol. Therefore, statistical methods are often used in hybrid solutions with threshold-based or simple anomaly classifiers; for example, studies using the combination of time interval and payload length are classic examples of statistical and machine learning (ML) hybrids (Han et al., 2020; Elsadig & Gafar, 2022).

Signature-Based Detection

Signature-based detection approaches rely on searching network traffic for predefined signatures or rule sets; these signatures can be specific header modifications, sequence number anomalies, signatures related to characteristic patterns in DNS FQDN formats, or unexpected application-level field combinations. Signature-based systems are particularly powerful in operational environments where known and reproducible covert channel techniques must be quickly detected, as signature matching provides low-latency and interpretable results. However, this approach has obvious limitations: attackers can easily bypass signatures through randomization, variation generation, the use of encrypted carriers, or protocol mimicry (Çavuşoğlu et al., 2020). Furthermore, signature-based methods are insufficient to discover new or derived covert mechanisms; therefore, ongoing maintenance of up-to-date signature databases and signature generation processes are required.

Machine Learning (ML) Based Approaches

Machine learning-based approaches have been one of the fastest-growing areas in the covert channel detection literature over the last decade. Studies in this area can be categorized into two main axes: (i) the classification of hand-crafted features via classical supervised learning methods (Random Forest, SVM, XGBoost) and (ii) automatic feature extraction from raw sequences/byte arrays via deep learning approaches (LSTM, CNN, Transformer) (Wang et al., 2019; Chen et al., 2021). Supervised models have been used in many studies, reporting strong performance, particularly in DNS tunneling and packet-length-based storage channels. On the other hand, RNN/LSTM-based sequential models have been found to be particularly effective in detecting FQDN or time-series-based covert modulations.

The main challenges of ML approaches are: the rarity of large, labeled, and representative datasets; data imbalance (rarity of covert examples); heterogeneity in carrier protocols (increased number of encrypted carriers with DoH/DoT/QUIC); and attacker adaptation (adversarial examples, data poisoning, impersonation). For these reasons, current research focuses on improving generalizability and robustness using techniques such as data augmentation, transfer learning, ensemble models, feature fusion (combining statistical and sequence-based features), and attention mechanisms (Elsadig & Gafar, 2022; Chen et al., 2021).

Deep Learning and Image-Representations

In recent years, deep learning models such as CNN, LSTM, and Transformer have gained popularity for classifying time series by converting them directly into two-dimensional matrices or images rather than raw sequences. This method allows for processing with both convolutional and sequential models, particularly by converting inter-arrival times, packet sizes, and other traffic attributes into a 2D "image-representation." For example, the SnapCatch approach (Al-Eidi et al., 2020) used a combination of image processing and machine learning to detect timing-based covert channels; this paradigm was later advanced with CNN-based localization methods and DL-based extensions. Models using Transformer and attention mechanisms can learn long-range dependencies in time series and capture more complex traffic patterns (Li et al., 2023). Although these approaches offer high detection accuracy, computational cost, model complexity, and latency in real-time applications must be considered. Additionally, the generalisability of DL-based systems is also determined by choices made in the generation of the DL-based system's training data, both in regard to its diversity and the amount of data chosen to generate it. Therefore, data preprocessing and augmentation techniques must be given careful consideration.

Protocol-Independent / Generic Frameworks

To address the limitations of traditional signature-based/protocol-specific methods, protocols and approaches have been created that are not dependent on any specific protocol type. These frameworks have less reliance on protocol-specific characteristics while providing better flexibility for detection among various protocols and within diverse types of the network. As an example, the types that were proposed by Ayub et al. (2019) were designed to find covert channels by identifying data characteristics that were not dependent upon any specific protocol type and then using general ML models to perform the discovery process.

In addition to ML-based methods, there are also non-ML-based methods that provide protocol-independent detection of covert communication channels. Examples of non-ML methods to perform protocol-independent detection of covert communication include perceptual hash techniques and timing pattern comparison methods, which can be applied to all different types of network traffic (Zhuang et al., 2024). Although protocol-independent detection techniques have a lot of promise for actual use, many of these frameworks will need to be validated thoroughly against actual heterogeneous traffic and evolving adversarial tactics. As such, it is likely that protocol-independent methods will be integrated with hybrid DL/ML-based approaches in the future to increase the overall performance of the detection.

Comparative Summary of Major Studies in The Last Decade

The data in Table 1 summarizes important studies conducted between 2015 and 2025 which detect covert channels. It also highlights the methodology used, the data utilized, and the results achieved by each of the studies. As seen in Table 1 over the past ten years significant changes to the methodologies used to detect covert channels have occurred over the last ten years. From the dates of

2016-2017, almost all studies focused on evaluating statistical variance, event correlation, and using signature-based techniques to analyze network flow, while there were several challenges with generalizability as a result of low computational complexity. However, machine learning techniques were incorporated beginning in 2018-2020, which increased the overall accuracy for detecting covert channels, and in one case exceeded 95% within multiple papers that incorporated SVM, Hopfield Neural Networks, Random Forest, and other algorithm designs which were not protocol-specific. Additionally, from 2021-2022, data leakage detection was analyzed using physical air-gaps, and ensemble machine learning methodologies provided accuracy levels for modern protocol-based and packet-based covert channels. The implementation of Transformer models, and robust channeling, for covert channels during 2023-2024 has provided significant increases in accurate detection and secure next-generation protocols (i.e. DNS, QUIC, and DTLS). More recently, AI-driven analysis of acoustic and optical side channels in air-gapped systems (2025) using deep neural networks has achieved detection accuracies above 95% and 90%, respectively, demonstrating a substantial improvement over classical threshold-based approaches. As seen in the table, there has been a clear shift from traditional means of detecting covert channels using statistical methods towards using multi-modal methods based on machine learning and deep learning.

Table 1: Leading studies on covert channel detection between 2015 and 2025

Year	Authors	Method / Model	Application Area	Dataset Used	Success (%) / Findings
2015	Guri et al.	Heat-based covert channel creation (BitWhisper); implicit communication model via thermal emissions and internal heat sensors	Covert channel creation and security analysis between air-gapped systems	Experiments on real hardware: different PC layouts (0–40 cm), various thermal sensor types, different CPU configurations and virtual machines	Bidirectional 1–8 bits/clock communication without extra hardware; ability to exfiltrate small data such as passwords; demonstrates bridging air-gapped systems via heat.
2016	Wang et al.	XSRM (SharedResourceMatrix) + event-association algorithm	Detection of multi-level and multi-format covert channels in cloud environment	Virtual machine event logs / shared resource metrics (cloud VM)	Reported as being able to "effectively identify real-time multi-level multi-format covert channels"
2017	Guri et al.	DiskFiltration – HDD-based acoustic covert channel using actuator-arm–generated audio frequencies	Air-gapped systems; data exfiltration via acoustic covert channels from speakerless computers	Experiments on various HDD models, chassis, and distances; transmitter on PCs and receiver on smartphones/PCs.	180 bits/min bitrate; successful exfiltration of sensitive data; HDD acoustic emissions proven modulated and detectable.
2019	Ayub et al.	Supervised machine learning (Decision Tree, KNN, Logistic Regression, SVM) — protocol-independent model	Detection of storage covert channels in IP, TCP, DNS protocols	Traffic containing storage covert channels over IP/TCP/DNS protocols	The phrase "excellent results for detection" was reported
2020	Saeli et al.	Machine learning (anomaly indicators + behavioral profiling)	Covert channel detection via DNS protocol	15-day experimental traffic monitoring system (DNS traffic + injection attack traffic)	"All malicious variants detected; low false positive rate" reported

2021	Shen et al.	EMLoRa – Electromagnetic covert channel using memory as a LoRa-like spread-spectrum radio	Air-gapped systems; long-range EM covert channels; bypassing shielding/Faraday cages; emission security attacks	Experiments on laptops, desktops, and servers at various distances and CPU loads; tested with and without shielding.	20× communication-range and up to 53 dB attenuation-resilience improvement; long-range decoding through shielding; detectors show limited effectiveness.
2023	Elsadig & Gafar	Ensemble machine learning (SVM + Random Forest + Naive Bayes base; combined with logistic regression meta-classifier)	Detection of "packet length" covert channels	A data set containing a created/covert channel on network traffic	The algorithm achieved an accuracy rate of 98.5%.
2023	Sun et al.	Transformer-based detection	DNS covert channel	Scientific facility network traffic (DNS)	High accuracy and low FPR reported
2024	Fischlin et al.	Robust channels—including sliding-window mechanism	QUIC and DTLS 1.3 recording layers	No specific real dataset; protocol and mathematical modeling	It has been shown that both protocols can provide "robust chosen-ciphertext security"
2025	Lesinskyi & Antoshchuk	Deep Neural Networks (AI-based signal analysis)	Detection of covert channels in air-gapped systems (acoustic & optical side-channel leakage)	Experimentally collected acoustic and optical emanation recordings from air-gapped hardware	Acoustic: >95% accuracy; Optical: >90% accuracy. AI models outperform classical threshold-based approaches and enable reliable detection even under noisy conditions.

APPLICATION AREAS AND CASE STUDIES

Timing Channels — SnapCatch and Variants

Through Timing Channels, it is possible to transfer data using the amount of time between each packet's transmission. SnapCatch (Al-Eidi et al., 2020) uses Machine Learning to classify Network Traffic based on Inter-Arrival Time Sequence. When these

sequences are represented, the resulting output can be visualized as a two-dimensional image (referred to as Image Representation). This not only helps to detect the presence of a covert channel, but also gives the tools needed for an organisation to correctly localise its data. SnapCatch popularized the image-based paradigm in the timing channel detection literature and paved the way for subsequent deep learning-based localization and convolutional methods.

DNS/DoH Covert Channels — LSTM and Transformer Applications

Many cybercriminals employ DNS Data Exfiltration as a form of cybercrime due to the fact that most DNS requests/responses do not go through any type of filtering and are sent/received using DNS over HTTPS or DNS over TLS, making it challenging to identify when a legitimate domain was being used for data exfiltration. Chen et al. (2021) have demonstrated the ability to identify DNS covert channels through the use of long short-term memory (LSTM) networks to process fully qualified domain names (FQDN) as a series of FQDN sequences. In addition, Transformer based methods developed between 2022-2024 continue to evolve, capturing long-range dependencies to further improve accuracy. Additionally, the CIRA-CIC-DoHBrw-2020 Dataset provided an important means of testing and validating these methods (MontazeriShatoori et al., 2020).

Packet-Length / Storage Channels and DGA-Related Methods

The use of DGA and DNS naming techniques to transfer data via packet headers is accomplished through the use of storage channels. The characteristics of storage channels can be identified by Wang et al.'s (2019) use of textual characteristics like TF-IDF and specificity scores to detect both DNS and DGA covert channels. These studies have demonstrated how effective it is to use traditional

methodologies for features engineering and ML classification for storage channels.

Air-Gap & Physical Channels

Air-gap systems rely on a variety of physical means to transmit data, including the brightness of LEDs, the speed of fans, electromagnetic leakage, and even ultrasonic signals. Network-based detection methods are typically not well-suited to these types of channels. Therefore, more emphasis is placed on using sensor-based monitoring and physical security to protect against data loss or unauthorized access through air-gap systems. Al-Eidi et al. (2020) used sensor-based analysis to study the detection of physical channels through the use of sensor-based analyses that identify the operation risks and vulnerabilities associated with air-gap systems.

DATASETS, EVALUATION METRICS, AND COMPARISON CHALLENGES

Throughout the entire research literature on covert channel detection, one huge limitation is the lack of reproducible results through standard methods of intercomparison. The primary reason for this limitation is that all of the datasets created and used for covert channel detection research are highly heterogenous and many times are specifically designed by the researcher for their own purposes. The datasets mentioned most frequently in the literature include the CIRA-CIC-DoHBrw-2020 dataset which was built to measure the DoH protocol and various special types of synthetic or controlled datasets created by different researchers (MontazeriShatoori et al. 2020). Because of the disparities in the types of testing environments that have been created by various researchers, such as the types of protocols and bandwidths used as well as the types of traffic volumes tested, it can be very difficult to do direct comparisons of methodologies produced by different researchers performing evaluations of datasets produced in various environments.

Traditional assessment measures for classification include accuracy, sensitivity, and specificity (i.e., precision/recall), as well as F1 and AUC scores, all widely used in peer-reviewed journal articles. Additionally, FPR (False Positive Rate) and detection latency are two key performance indicators for the use of detection systems in practice. A high FPR within operational systems results in an equally high likelihood that an operator will tune out responses to an excessive number of false alarms; both wasted resources and ignored critical alerts can occur. Therefore, utilizing both traditional classification metrics as well as operational metrics when evaluating covert channel detection systems is critical.

In addition, a variety of datasets and evaluation metrics have been used to evaluate covert channel detection methods in the literature. Unfortunately, using the various datasets and metrics makes it difficult to create a communized comparison framework. To facilitate future research in this area, it will be necessary to expand on existing benchmark datasets to include a wider range of heterogeneous traffic types and to create and agree upon a standardized set of metrics.

OPEN PROBLEMS (RESEARCH GAPS) AND RECOMMENDATIONS

In recent years, there has been a considerable amount of research conducted on covert channels. However, there are still major challenges that must be addressed. As many of the studies published in the literature typically focus on a particular type of network environment or protocol without providing any benchmark data for comparison or with generalization of their model to other environments, there is little knowledge regarding how well these solutions work in different environments. Additionally, there are still challenges to be overcome with regards to the use of Deep Learning Based Models in operational environments, including fast and low

latency response time, resilience to adversarial attacks, and the ability to evaluate multiple data types at once. As such, current open questions and potential areas for future research can be found in Table 2.

Table 2: Research Gaps and Recommendations

Open Problem / Research Gap	Definition	Recommended Approach / Solution Proposal	Source / Case Study
Generalizability / Domain Shift	A model trained in a data center or ISP environment may experience performance loss in different network environments	Domain adaptation, transfer learning, and cross-domain training strategies	Elsadig & Gafar, 2022
Lack of Benchmark and Labeled Data	Heterogeneous and lack of up-to-date covert-channel datasets	Creation of comprehensive benchmark datasets (including DoH, DoT, QUIC, IoT protocols)	Elsadig & Gafar, 2022; MontazeriShatoori et al., 2020, 2020; Chen et al., 2021
Real-time, Edge-aware Models	DL-based models cannot run with low latency on operational and edge devices	Model compression, pruning, distillation, hardware acceleration	Abualghanam et al., 2023; Elsadig & Gafar, 2022
Adversarial Resistance	The probability of an attacker deceiving an ML-based defense	Adversarial test-suits, robust training, security testing	Fischlin et al., 2024; Elsadig & Gafar, 2022
Multi-modal / Hybrid Approaches	Limited performance of single-model approaches	Statistical pre-filter → lightweight ML → heavy DL chaining when necessary; integrated analysis of different data types	Al-Eidi et al., 2020; Darwish et al., 2019; Elsadig & Gafar, 2022

Table 2 outlines the major open problems in the field and summarizes how the existing literature has addressed them through proposed solutions and illustrative case studies. For instance, Elsadig & Gafar (2022) provide a notable example for the issue of 'Generalizability/Domain Shift.' The problem of insufficient benchmark data is discussed both through available datasets (MontazeriShatoori et al., 2020) and studies such as Chen et al. (2021). The need for real-time and edge-aware models is emphasized in the works of Abualghanam et al. (2023) and Elsadig & Gafar (2022), while adversarial resilience is explored by Fischlin et al.

(2024). Multimodal or hybrid approaches are represented in the methods of Al-Eidi et al. (2020), Darwish et al. (2019), and Elsadig & Gafar (2022). Overall, Table 2 serves as a concise reference point for identifying current gaps in the literature and highlights several directions that future research may need to pursue.

DISCUSSION AND CONCLUSION

Over the recent decade, there has been an overwhelming increase in the use of different types of methods for covert channel detection and protection research, which have included an increasing focus on using ML and DL based approaches. Many of the methods that have arisen for timing-based channels rely on converting Inter-Arrival Times (IAT) to images and conducting a CNN based classification of the traffic through methods like SnapCatch & its derivatives developed by Al-Eidi et al. (2020) which have allowed the ability to localise covert data segments, resulting in the widespread use of time-series based DL applications.

When it comes to DNS & DoH protocol-based covert channels have been researched successfully using LSTM, Transformer-based methods (Chen et al., 2021; Li et al., 2023) which has shown good success in carrying out the task of detecting long-range dependencies and FQDN-based covert modulations, as well as being able to provide a holistic view of the security landscape when they were developed alongside a protocol-independent detection framework (Ayub et al., 2019; Zhuang et al., 2024). Furthermore, packet-length and storage channels (as well as DGA-related methods) have also been detected successfully using NLP-based features and ML classification methods (Wang et al., 2019).

Nevertheless, there are still significant limitations in the foundation and methodology of the research literature. Some of the major obstacles include: No common dataset; Model generalization across heterogeneous network environments; Optimizing deep

learning (DL) models for real-time operational use; Integrating multimodal data; and Resilience to adversarial attacks. These gaps affect the ability to compare research across different institutions, as well as the ability to apply practical security solutions to real-world problems (Elsadig & Gafar, 2022; Fischlin et al., 2024).

The purpose of this review is to investigate systematically the high-quality studies published between 2015 and 2025 that focused on detecting the use of timing channels, storage channels, protocol misuse channels, and physical covert channels, from the perspective of statistical, signature-based, or ML-DL detection methods. This review discusses, in detail, all of the challenges and limitations currently facing researchers regarding datasets, evaluation metrics, and the challenges of comparing various detection methodologies.

In summary, the research literature has made considerable progress in detecting covert channels but many problems remain in the academic community that require solutions. Providing an easily usable reference guide for future research, will allow researchers to investigate further what currently exists in the literature and maintain current findings.

Consequently, it is essential that researchers concentrate their future efforts on acquiring open access and heterogeneous datasets, the use of domain-adapting and transfer-learning techniques, developing adversarially robust, and lightweight edge-protecting DL/ML models to enhance both academic and operational performance in detecting covert channels.

REFERENCES

Abualghanam, O., Alazzam, H., Elshqeirat, B., Qatawneh, M., & Almaiah, M. A. (2023). Real-time detection system for data exfiltration over DNS tunneling using machine learning. Electronics, 12(6), 1467. https://doi.org/10.3390/electronics12061467

Al-Eidi, S., Darwish, O., Chen, Y., & Husari, G. (2020). SnapCatch: Automatic detection of covert timing channels using image processing and machine learning. IEEE Access, 9, 177–191. https://doi.org/10.1109/ACCESS.2020.3046234

Al-Eidi, S., Darwish, O., & Chen, Y. (2020). Covert timing channel analysis either as cyber attacks or confidential applications. Sensors, 20(8), 2417. https://doi.org/10.3390/s20082417

Ayub, M. A., Smith, S., & Siraj, A. (2019). A protocol independent approach in network covert channel detection. In 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (pp. 165–170). IEEE. https://doi.org/10.1109/CSE/EUC.2019.00040

Bell, D. E., & La Padula, L. J. (1976). Secure computer system: Unified exposition and multics interpretation. MITRE Technical Report, MTR-2997.

Caviglione, L. (2021). Trends and challenges in network covert channels countermeasures. Applied Sciences, 11(4), 1641. https://doi.org/10.3390/app11041641

Çavuşoğlu, İ. G., Alemdar, H., & Onur, E. (2020). Covert channel detection using machine learning. In 2020 28th Signal Processing and Communications Applications Conference

(SIU) (pp. 1–4). IEEE. https://doi.org/10.1109/SIU49456.2020.9302098

Chen, S., Lang, B., Liu, H., Li, D., & Gao, C. (2021). DNS covert channel detection method using the LSTM model. Computers & Security, 104, 102095. https://doi.org/10.1016/j.cose.2020.102095

Darwish, O., Al-Fuqaha, A., Brahim, G. B., Jenhani, I., & Vasilakos, A. (2019). Using hierarchical statistical analysis and deep neural networks to detect covert timing channels. Applied Soft Computing, 82, 105546. https://doi.org/10.1016/j.asoc.2019.105546

Elsadig, M. A., & Gafar, A. (2022). Covert channel detection: Machine learning approaches. IEEE Access, 10, 38391–38405. https://doi.org/10.1109/ACCESS.2022.3164392

Elsadig, M. A., & Gafar, A. (2023). An ensemble model to detect packet length covert channels. International Journal of Electrical & Computer Engineering, 13(5). http://doi.org/10.11591/ijece.v13i5.pp5296-5304

Fischlin, M., Günther, F., & Janson, C. (2024). Robust channels: Handling unreliable networks in the record layers of QUIC and DTLS 1.3. Journal of Cryptology, 37(2), 9. https://doi.org/10.1007/s00145-023-09489-9

Guri, M., Monitz, M., Mirski, Y., & Elovici, Y. (2015). Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In 2015 IEEE 28th Computer Security Foundations Symposium (pp. 276-289). IEEE. https://doi.org/10.1109/CSF.2015.26

Guri, M., Solewicz, Y., Daidakulov, A., & Elovici, Y. (2017). Acoustic data exfiltration from speakerless air-gapped

computers via covert hard-drive noise ('DiskFiltration'). In European symposium on research in computer security (pp. 98-115). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-66399-9_6

Han, J., Huang, C., Shi, F., & Liu, J. (2020). Covert timing channel detection method based on time interval and payload length analysis. Computers & Security, 97, 101952. https://doi.org/10.1016/j.cose.2020.101952

Lampson, B.W. (1973). A Note on the Confinement Problem. Communications of the ACM. Oct, 10, 613-615.

Lesinskyi, V., & Antoshchuk, Y. (2025). Analysis and investigation of artificial intelligence-enabled attacks on air-gapped information systems. In Seventeenth International Conference on Correlation Optics, 13813, 534-538). https://doi.org/10.1117/12.3092641

Li, Z., Chen, Y., Teng, Z., & Huang, X. (2023). Contra: A covert timing channel detection approach for little covert information in a network. In Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things (pp. 614–620). https://doi.org/10.1145/3603781.36038

MontazeriShatoori, M., Davidson, L., Kaur, G., & Lashkari, A. H. (2020). Detection of DoH tunnels using time-series classification of encrypted traffic. In 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech) (pp. 63–70). IEEE. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026

Saeli, S., Bisio, F., Lombardo, P., & Massa, D. (2020). DNS covert channel detection via behavioral analysis: A machine learning approach. arXiv preprint arXiv:2010.01582. https://doi.org/10.48550/arXiv.2010.01582

Shen, C., Liu, T., Huang, J., & Tan, R. (2021). When LoRa meets EMR: Electromagnetic covert channels can be super resilient. In 2021 IEEE Symposium on Security and Privacy (SP) (pp. 1304-1317). IEEE. https://doi.org/10.1109/SP40001.2021.00031

Sun, Q., Liu, J., Wang, J., Yan, T., Ana, D., & Qia, F. (2023). Transformer-based detection method for DNS covert channel. In Workshop, ISGC & HEPiX2023 (Vol. 19, p. 31).

Wang, L., Liu, W., Kumar, N., He, D., Tan, C., & Gao, D. (2016). A novel covert channel detection method in cloud based on XSRM and improved event association algorithm. Security and Communication Networks, 9(16), 3543–3557. https://doi.org/10.1002/sec.1560

Wang, Z., Dong, H., Chi, Y., Zhang, J., Yang, T., & Liu, Q. (2019). DGA and DNS covert channel detection system based on machine learning. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering (pp. 1–5). https://doi.org/10.1145/3331453.336166

Zhuang, X., Chen, Y., & Tian, H. (2024). A generalized detection framework for covert timing channels based on perceptual hashing. Transactions on Emerging Telecommunications Technologies, 35(5), e4978. https://doi.org/10.1002/ett.4978

BÖLÜM 7
PSO-Integrated K-Means Algorithm for Robust Initialization and Fast Convergence

Özge ASLAN YILDIZ[1]

1-Introduction

The performance of the K-means clustering algorithm is significantly affected by the choice of the initial cluster centers. If the algorithm starts from poor initial positions, it may converge to suboptimal solutions and require more iterations to stabilize. In practical applications, this sensitivity is a major limitation, especially for high-dimensional data or datasets with overlapping clusters. To address this issue, this study proposes a hybrid strategy in which Particle Swarm Optimization (PSO) is used to determine high-quality initial cluster centers for K-means. The PSO algorithm, with its global search and optimization capabilities, provides a new approach to improve the shortcomings of traditional K-means algorithms in cluster center initialization. PSO performs a global search in the solution space by evolving a population of candidate solutions guided by personal and global best experiences. By using PSO to obtain promising cluster-center configurations, K-means can start from a more informative point and complete its local refinement with fewer iterations. As a result, the proposed approach aims to reduce the number of K-means iterations, improve convergence speed, and increase robustness against poor initializations.

Recent advances in data collection and sensing technologies have led to increasingly large, high-dimensional, and heterogeneous datasets, making robust and efficient clustering more critical than ever. In this context, researchers have widely explored hybrid approaches that combine the fast local refinement of K-means with the global search capability of meta-heuristics such as Particle Swarm Optimization. The literature consistently reports that PSO assisted K-means variants can mitigate sensitivity to initialization, improve clustering quality, and reduce the number of iterations, especially in complex search spaces where standard K-means is prone to poor local optima. Moreover, these hybrid methods have been applied in many current domains, including large-scale data processing, image segmentation, customer analytics, multi-robot task allocation, and digital governance. These applications show their practical value and growing research interest.

[1] Asst. Prof. Dr., Erzincan Binali Yildirim University, Faculty of Engineering and Architecture, Department of Computer Engineering, Erzincan, Turkey, Orcid: 0000-0001-7688-9326

The number of publications regarding hybrid K-means and PSO in Web of Science (WOS) between 2010 and 2025 is shown in Figure 1. To obtain Figure 1, the WOS database was searched with the query "(TS="K means" OR TS=K-Means) AND (TS="PSO" OR TS="particle swarm optimization")". An examination of Figure 1 shows that interest in this topic has increased in recent years.

Ahmedyfard and Modares (2008) made the K-means algorithm run faster by performing a certain number of iterations of the clustering phase using PSO and the remaining iterations using the K-means algorithm. Ghali et al. (2008) proposed the EPSO method, which they used in clustering problems by exponentially applying the PSO algorithm. This method, where PSO parameters are defined exponentially rather than linearly, yielded more successful results than the PSO algorithm. Aljarah and Ludwig (2012) published their work using the PSO algorithm in combination with the MapReduce method for clustering large datasets. The study showed that this method, called MR-CPSO, increased clustering speed by performing clustering in parallel. Yi et al. (2006) proposed the Fuzzy PSO method, which uses fuzzy logic to perform clustering on images. This method, which probabilistically assigns membership functions to the data, was able to successfully cluster images. Nikham et al. (2008) published their study proposing the PSO-ACO method, which combines PSO with artificial bee colony methods for clustering. In this method, the best position of each particle is determined using the artificial bee colony method, and the PSO algorithm is used for clustering.

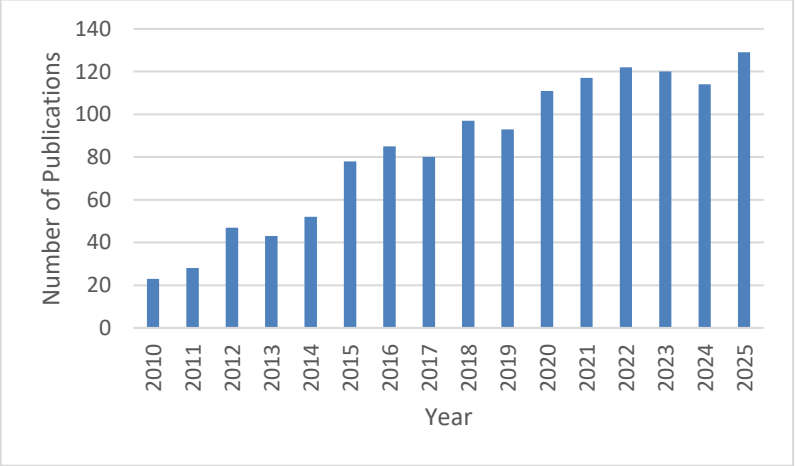


Figure 1: The number of papers for hybrid K-means and PSO

Masoud (2023) investigated a hybrid K-means and PSO approach for optimizing e-scooter charging routes, using clustering to group nearby scooters and PSO to search for efficient routes within/among these groups. In benchmarking experiments, the hybrid K-means/PSO led in roughly 52% of scenarios and uniquely exceeded Tabu Search in one case. Zarei & Nickfarjam (2023) proposed a hybrid K-means and PSO algorithm for image segmentation. PSO selects the best K value and finds better initial centroids using Structural Similarity Index (SSIM) as the fitness function. Experiments show that it performs better than standard K-means. Li et al., (2023) proposed IHPSO-KM, a customer segmentation approach that uses an improved hybrid PSO (IHPSO) to optimize the initial cluster centers of K-means. IHPSO enhances search accuracy and avoids local optima through parameter tuning and

adaptive selection, crossover, and mutation operations. Experiments on benchmark functions and five UCI datasets showed that IHPSO and IHPSO-KM outperform several PSO- and GA-based baselines, and the method proves effective in real customer segmentation. Yuan et al., (2024) proposed a multi-robot task assignment method that combines K-means++ and PSO algorithm. First, K-means++ clusters task targets, and PSO assigns these clusters to robots using robot–cluster distance. PSO then optimizes the task order within each cluster, and ROS-based simulations and real experiments show improved collaborative efficiency over auction-based and non-clustering PSO baselines. Yue et al., (2025) introduced PSO-KM, a hybrid PSO and K-means algorithm for clustering high-dimensional resident profile data to support digital governance. Using 2023 community management records, it updates centroids via PSO's global search and K-means' local refinement, achieving strong clustering quality.

2. Matherial and Methods

2.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based optimization method introduced by Kenedy and Eberhart (1995), inspired by the collective behavior of bird flocks and fish schools. It belongs to the broader family of swarm intelligence algorithms. Observations of animals moving in groups suggest that simple social interactions can lead to efficient search behavior, particularly in tasks related to foraging and safety. PSO is built on this idea by allowing individuals to share social information during the search process. The search process is performed as many times as the number of generations. In PSO, each candidate solution is called a particle, and the set of particles forms a swarm. The algorithm iteratively updates particle positions over a number of iterations. (Each individual is called a particle, and the population of particles is called a swarm. Each particle adjusts its position to the best position in the swarm, using its previous experience. PSO is fundamentally based on bringing the position of individuals in the swarm closer to the position of the individual in the swarm with the best position. This rate of convergence is a randomly occurring state, and often, individuals within the swarm improve their position in their new movements, and this process continues until the goal is reached (Kennedy & Eberhart, 1995; Poli et al., 2007).

The flowchart of the PSO algorithm is shown in Figure 2. In PSO, particles are encouraged to explore different regions of the search space throughout the optimization process. Each particle is associated with a position vector and a velocity vector. At each iteration, the velocity is updated based on the particle's distance to its own best-known position (personal best) and the swarm's best-known position (global best). The updated velocity is then added to the current position to generate a new candidate solution. If the new solution improves the particle's personal best or the swarm's global best, these best values are updated accordingly. (Kennedy & Eberhart, 1995; Poli et al., 2007; Özsağlam & Çunkaş, 2008). The steps of the PSO algorithm are as follows:

i. An initial swarm is generated. The initial positions and velocities of each particle in the swarm are randomly assigned.
ii. The fitness value of the particles is calculated according to the given objective function.

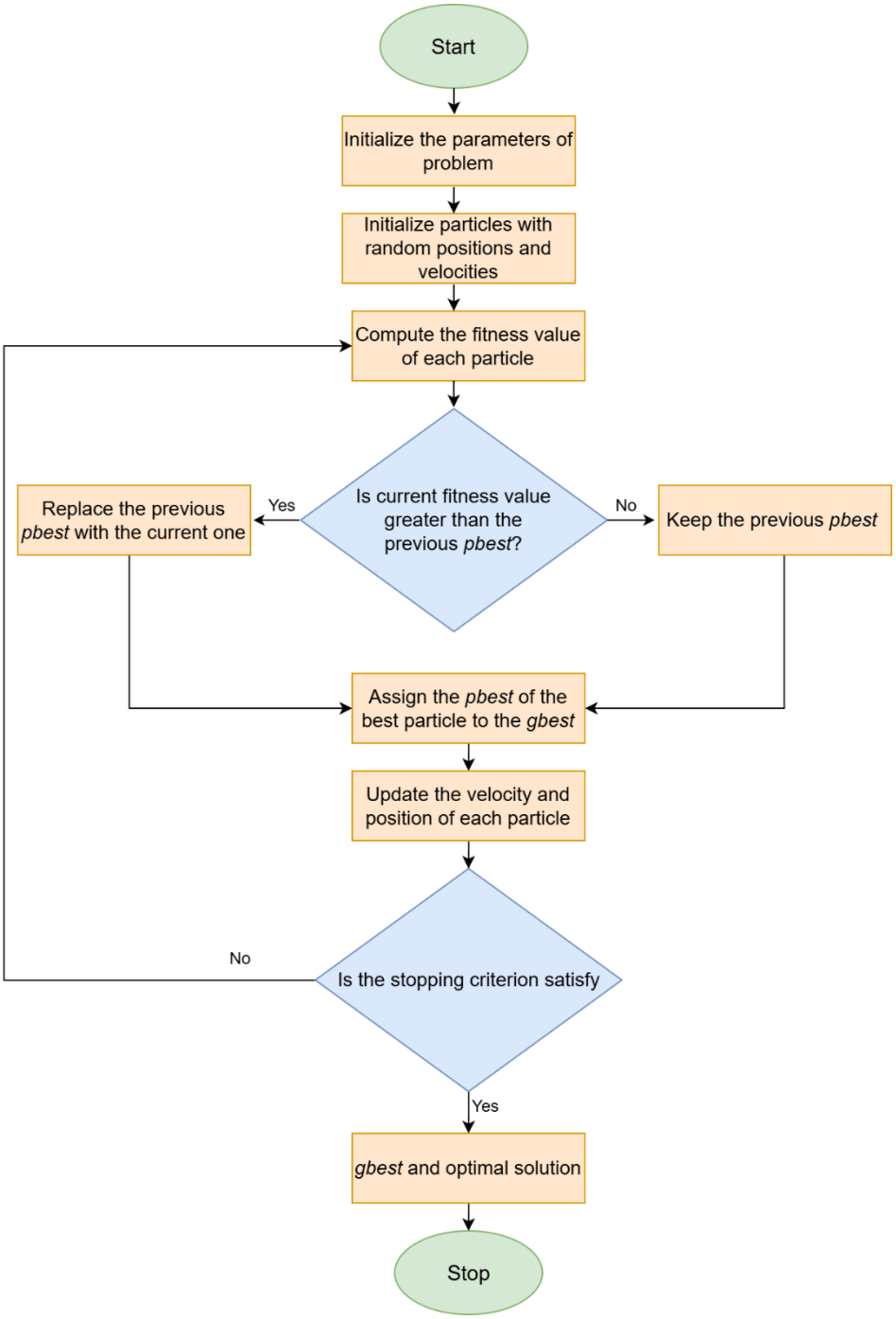


Figure 2: The flowchart of the PSO algorithm

iii. The best local value of the particle is found. The particle value calculated in the previous step is compared with the local best value stored in the particle's memory, based on their fitness values. If the fitness value of the particle found in the previous step is better than the fitness value of the current local best particle, the local best value is updated. Otherwise, it remains the same.

iv. The global best value of the particle is found. If the fitness value of the best particle in the swarm calculated in step ii is better than the best global value stored in the swarm memory, the global best value is updated. Otherwise, it remains the same.

v. New velocity (V) and position vectors (X) are calculated for the particles within the

search space. This process is performed separately for each particle. Learning factors (c_1, c_2) enable the search space to be around the local best value of the particle (c_1) and the global best value (c_2). Random numbers (r_1, r_2) are uniformly distributed random numbers in the interval [0 1]. The constant of inertia (s) is the coefficient used to control the velocity of the particle.

$$V_i^t = w \times V_i^{t-1} + c_1 \times r_1 \times (Y_i - X_i^{t-1}) + c_2 \times r_2 \times (G - X_i^{t-1}) \quad (1)$$

$$X_i^t = X_i^{t-1} + V_i^t \quad (2)$$

vi. If the stopping criterion (fitness function) is met, the algorithm terminates. If this criterion is not met, the process returns to step ii.

2.2. K-Means Clustering

The K-means method, the most widely used clustering method, was proposed by S. Lloyd in 1957. Easier to implement than other methods, the K-means method allows each data point to belong to only one cluster. In the algorithm, each cluster is represented by its cluster center. The distance of a new data point from these cluster centers determines which cluster it belongs to. The parameter k, which gives the algorithm its name, is an integer representing the number of clusters. This parameter is determined before the algorithm starts and remains constant until the clustering process is complete (Jain et al., 1999). Let N be D-dimensional datasets, $\{x_1, x_2, \dots, x_n\}$. We want to divide this data into k clusters. In this case, there are k D-dimensional cluster centers, $\{u_1, u_2, \dots, u_k\}$. Let each data point be represented by a cluster label, $j = 1,2, \dots, k$, as c_j. The algorithm steps are as follows:

1. Choose the number K that represents the number of sets.
2. Randomly select k D-dimensional cluster centers.
3. Repeat steps 4 and 5 until the cluster centers are fixed or shifted below a certain threshold.
4. For each element $j_n = \underset{k}{\text{Argmin}} \sqrt[2]{\sum_{i=1}^{D}\left(x_i - u_i^k\right)^2}$
5. For each cluster $u_k = \frac{\{\sum_{i=1}^{N}(j_i == k) \times x_i\}}{\{\sum_{i=1}^{N}(j_i == k) \times 1\}}$

The algorithm begins by randomly determining k cluster centers. Then, the distance of each data point to the cluster centers is measured using distance metrics. Euclidean distance is generally used for distance measurement. This process is shown in Step 4. The distance of each data point to all clusters in D dimensions is calculated individually. Based on the results, the data is assigned to the cluster closest to it. At the end of this step, the data assigned to the clusters changes. In Step 5, the cluster centers are recalculated. This calculation is found by dividing the sum of the data values assigned to the cluster by the number of data points. These processes are repeated until the cluster centers have changed completely or until the

specified number of cycles is reached. Figure 3 shows the working steps of the K-means algorithm for k=2 with an aligned (0 mean and unit standard deviation) Old Faithful dataset. This two-dimensional dataset has variables representing blast and wait times.

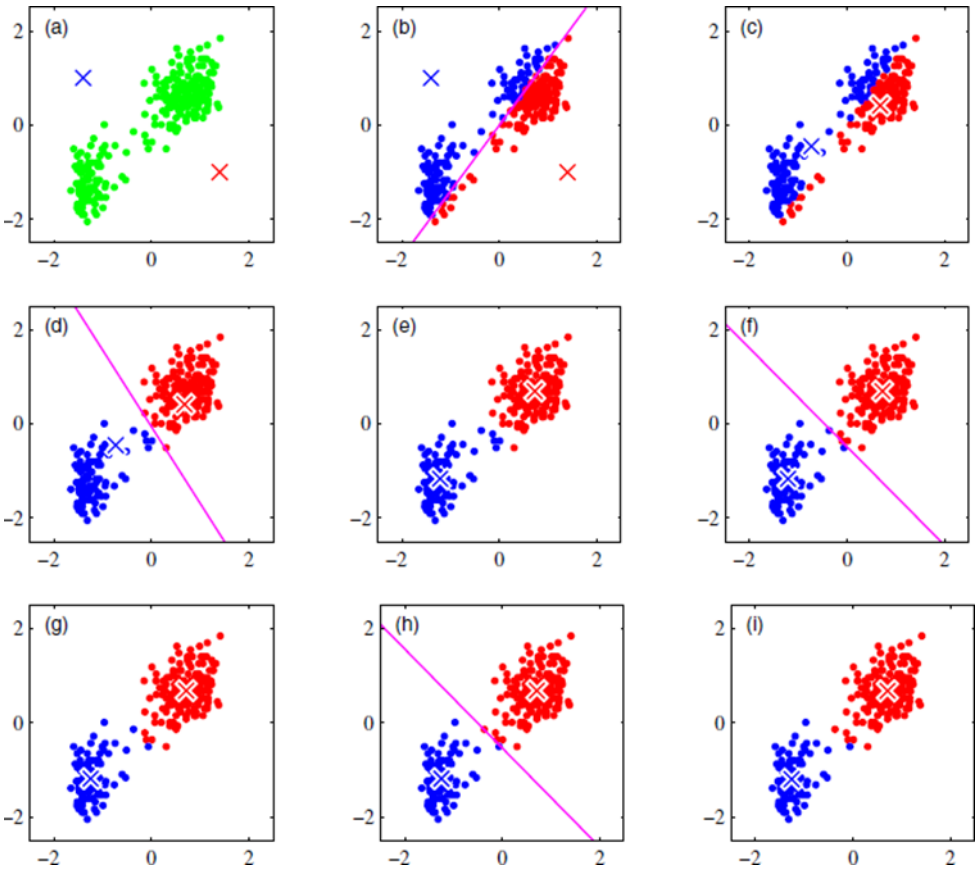


Figure 3: K-means clustering cycle. a) First step. b) First loop. The distance to the cluster centers is calculated for each data point, and the data points are assigned to clusters where they are closest. c) New cluster centers are calculated. d, e, f, g, h, i) The loop continues until the cluster centers are fixed

2.3. PSO-Integrated K-Means

Determining the initialization positions of cluster centers in the K-means algorithm increases the clustering speed and success of the algorithm. In the classical K-means algorithm, after determining the number of clusters, clustering begins by positioning the cluster centers at random positions in the space where the data is located. As seen in the example in Figure 4, initialization of the cluster centers in poor positions relative to the data positions can reduce the clustering success of the algorithm. For this reason, an optimization problem was created using the particle swarm algorithm to determine the level of cluster initialization positions of the K-means algorithm. An objective function that determines the quality of cluster positions was created and optimized with the PSO algorithm. In the objective function seen in Equation 3, the distance of each data to each particle is calculated and the data is assigned to the particle with the minimum distance value.

In Equation 3, the data are expressed as $x\{x_1, x_2, \dots, x_n\}$, the particles as $\{u_1, u_2, \dots, u_k\}$, and D represents the data dimension. In the study, Euclidean distance was used for distance calculation in the K-means algorithm. The sum of the distances obtained for all data represents the objective function value. The algorithm terminates its operation when the change in the objective function falls below a certain value or after a certain number of iterations. The particle with the best value among the obtained particles forms the initialization positions of the K-means clustering algorithm.

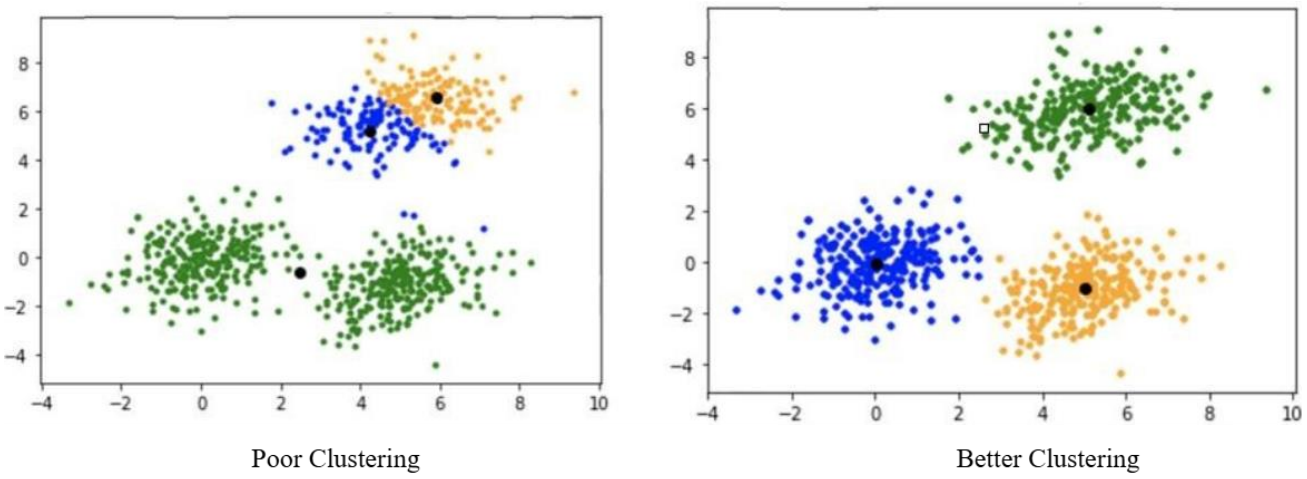


Figure 4: Sample Clustering Comparison.

The steps of the method developed within the scope of the study are given below.

i. Randomly determine the initial position and velocity of each particle. Here, each particle represents a potential cluster center.

ii. Calculate the objective function for generated particles. The objective function is (Fitness Function) value for the shown in Equation 3.

$$F(i) = \sum_{x=1}^{k} \left(\underset{k}{\text{Argmin}} \sqrt[2]{\sum_{i=1}^{D} \left(x_i^k - u_i\right)^2} \right) \tag{3}$$

iii. If the specified number of iterations is reached, or if the rate of change falls below a certain value, proceed to step v; otherwise, proceed to step iv.

iv. iv. Determine the global best state of the particle and the local best state of each particle and update the state of the particles using (1) and (2) and proceed to step ii.

3. Experimental Results

In the application carried out within the scope of the study, the values of the parameters used in the PSO algorithm were determined as $w = 0.8$, $c1 = 2$, $c2 = 2$ and $r1$, $r2 \in [0,1]$. Here, $r1$, $r2$ are two constants randomly determined between 0 and 1. w is the particle's old velocity retention coefficient and is generally determined between 0.5 and 1. c1 and c2 are used to determine the effect of global and local best values on the particle's new

velocity.

The Old Faithful dataset (Wanniarachchi., 2017) was used to test the developed method. This two-dimensional dataset has variables representing eruption and waiting times. The data were normalized between 0 and 1. Tests were performed using MATLAB R2024b on a computer with an i7-13700K 3.4GHz processor and 16 Gb RAM. The initial cluster positions obtained from the test, where the number of particles was set to 10 and the number of iterations to 50 for the dataset with two clusters, are shown in Figure 5. Looking at the results, it is seen that the method developed within the scope of the study can predict the initial positions of the K-means clustering algorithm quite close to the actual cluster centers. According to these obtained initial positions, the K-means algorithm will both reach a result faster and will be able to cluster the data correctly with a higher probability compared to random initial positions.

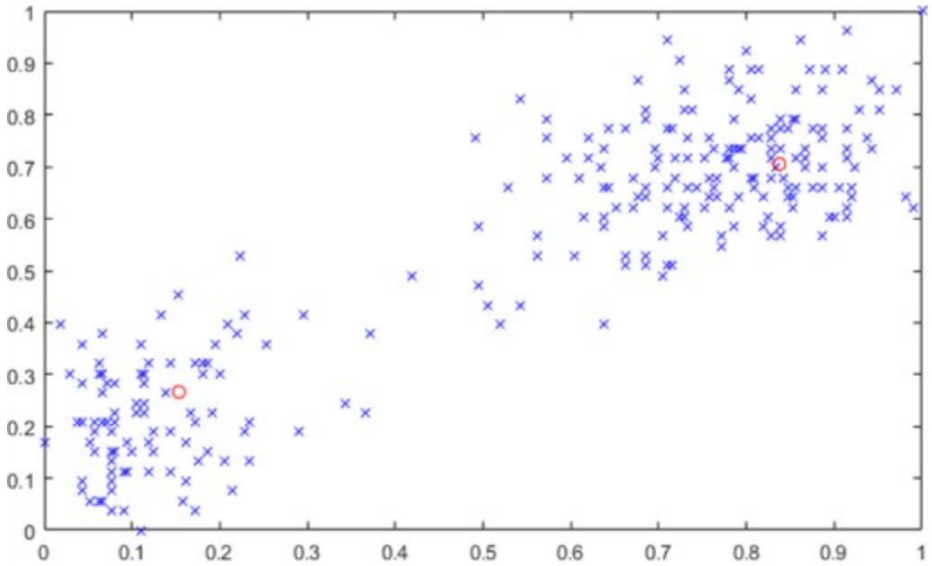


Figure 5: The initialization positions found by the PSO algorithm after 50 iterations for 10 particles. Red circles indicate the initialization positions.

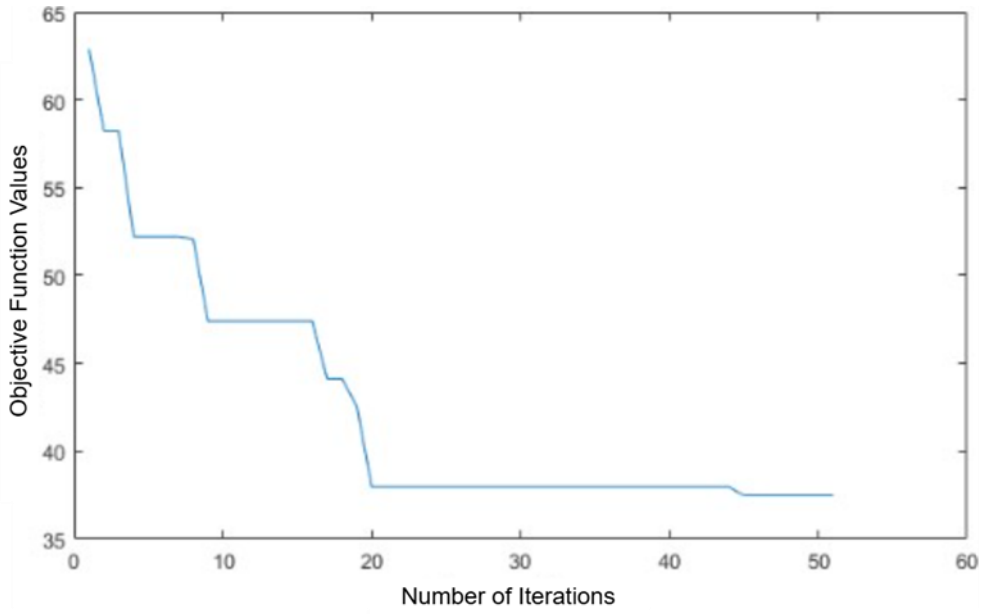


Figure 6: The objective function values obtained throughout the iterations for 50 iterations for 10 particles.

Figure 6 shows the graph of the objective function values obtained for each iteration of the test shown in Figure 5. Accordingly, the PSO-based method created has achieved its goal of minimizing the objective function value in each iteration. In the experiment, after the 20th iteration, the minimization of the objective function value becomes almost zero. In this case, rather than a fixed number of iterations, stopping the algorithm by looking at the difference between the objective function values can provide an advantage in terms of runtime.

4. Conclusion

This study investigates how improving the initialization stage can enhance the effectiveness of K-means clustering. In the classical K-means algorithm, the initial cluster centers are typically selected at random. However, choosing more suitable initial centers for a given dataset can reduce runtime and increase the likelihood of obtaining high-quality clustering results. In this work, the selection of initial cluster centers is formulated as an optimization problem, and Particle Swarm Optimization is employed to search for promising initializations through an objective function consistent with the K-means criterion. Experiments on the two-dimensional, normalized Old Faithful dataset, using fixed parameter settings, demonstrate that the proposed approach can successfully improve the initialization process and support efficient clustering. Therefore, the developed method can be integrated with K-means and applied to a range of clustering tasks.

Despite these encouraging results, this study has several limitations. First, the experimental evaluation was conducted on a single dataset with two-dimensional features, which may not fully reflect the behavior of the method on high-dimensional or more complex real-world data. Second, the PSO and K-means settings were kept fixed rather than being automatically tuned, and the analysis did not include extensive comparisons across multiple datasets and clustering validity indices. As future work, we plan to evaluate the approach on

diverse benchmark and real-world datasets, extend the analysis to higher-dimensional scenarios and different values of K, and develop an adaptive parameter-tuning strategy that selects PSO hyperparameters based on dataset characteristics. These improvements are expected to further enhance clustering performance, robustness, and general applicability.

5. References

Ahmadyfard, A., & Modares, H. (2008, August). Combining PSO and k-means to enhance data clustering. In *2008 international symposium on telecommunications* (pp. 688-691). IEEE.

Aljarah, I., & Ludwig, S. A. (2012, November). Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In *2012 fourth world congress on nature and biologically inspired computing (NaBIC)* (pp. 104-111). IEEE.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.

Ghali, N. I., El-Dessouki, N., Mervat, A. N., & Bakrawi, L. (2009). Exponential particle swarm optimization approach for improving data clustering. *International Journal of Electrical, Computer, and Systems Engineering*, *3*(4), 208-212.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264-323.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). ieee.

Li, Y., Qi, J., Chu, X., & Mu, W. (2023). Customer segmentation using K-means clustering and the hybrid particle swarm optimization algorithm. *The computer journal*, *66*(4), 941-962.

Masoud, M. (2023). A hybrid K-means and particle swarm optimization technique for solving the rechargeable E-scooters problem. *IEEE Access*, *11*, 132472-132482.

Niknam, T., Nayeripour, M., & Firouzi, B. B. (2008, December). Application of a new hybrid optimization algorithm on cluster analysis. In *Proceedings of world academy of science, engineering and technology* (Vol. 36, p. 599).

Özsağlam, M. Y., & Çunkaş, M. (2008). Optimizasyon problemlerinin çözümü için parçaçık sürü optimizasyonu algoritması. *Politeknik Dergisi*, *11*(4), 299-305.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm intelligence*, *1*(1), 33-57.

Wanniarachchi, J., “Old Faithful” Kaggle, ver. 1, 201. [Online]. Available: https://www.kaggle.com/datasets/janithwanni/old-faithful/data Accessed: Dec. 16, 2025.

Yi, W., Yao, M., & Jiang, Z. (2006, November). Fuzzy particle swarm optimization clustering and its application to image clustering. In *Pacific-Rim Conference on Multimedia* (pp. 459-467). Berlin, Heidelberg: Springer Berlin Heidelberg.

Yuan, Y., Yang, P., Jiang, H., & Shi, T. (2024). A Multi-Robot Task Allocation Method Based on the Synergy of the K-Means++ Algorithm and the Particle Swarm

Algorithm. *Biomimetics*, *9*(11), 694.

Yue, H., Zhang, H., & Dai, Y. (2025). Application of PSO-integrated K-means algorithm in resident digital portrait classification. *PLoS One*, *20*(8), e0329123.

Zarei, M. A., & Nickfarjam, A. M. (2023, February). PSO-based procedure to find number of clusters and better initial centroids for K-means algorithm: image segmentation as case study. In *2023 6th International Conference on Pattern Recognition and Image Analysis (IPRIA)* (pp. 1-4). IEEE.

BÖLÜM 8

ARTIFICIAL INTELLIGENCE–ENABLED THREAT DETECTION AND RISK MANAGEMENT IN CYBERSECURITY

ERCAN ERKALKAN[1]

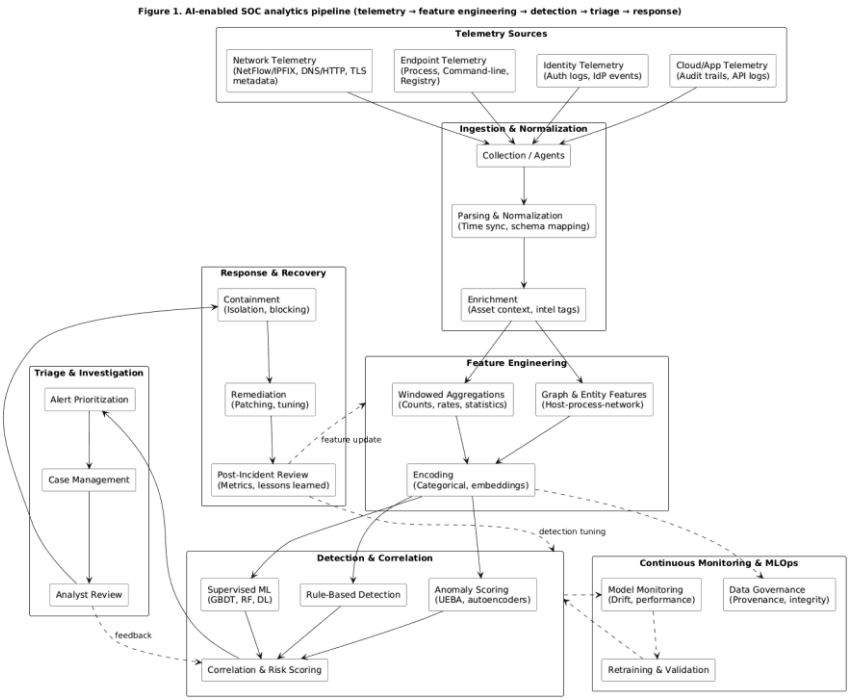
Introduction

Cybersecurity has become a critical enabler of digital transformation across cloud computing, Internet of Things (IoT), and 5G/6G infrastructures. Expanding attack surfaces, heterogeneous assets, and increasing operational complexity have reduced the effectiveness of purely signature-driven and rule-based security controls. Contemporary threat environments demonstrate rapid adaptation cycles, including exploitation automation, supply-chain compromise, and multi-stage campaigns mapped to structured adversary behaviors. Strategic threat knowledge bases such as MITRE ATT&CK provide a common language for describing tactics and techniques, supporting detection engineering and risk-informed defense planning(MITRE, 2025). Industry and regional threat analyses emphasize increasing scale and diversity of incidents,

[1] Dr. Öğr. Üyesi, Marmara Üniversitesi, Yapay Zeka Operatörlüğü Programı Orcid: 0000-0001-9259-7112

reinforcing the need for faster detection and response. The Verizon Data Breach Investigations Report (DBIR) synthesizes evidence from large incident and breach datasets to support data-driven prioritization of defensive controls (Verizon, 2024). The ENISA Threat Landscape provides an EU-oriented view of evolving threat types and trends, supporting strategic alignment of cybersecurity programs with observed adversarial activity (ENISA, 2024). Within this context, artificial intelligence (AI) and machine learning (ML) methods have become central to anomaly detection, behavioral analytics, and event correlation. However, the integration of ML components into security pipelines introduces additional risk vectors such as adversarial evasion and poisoning, as well as governance challenges related to transparency, accountability, and continuous monitoring (Vassilev et al., 2025).

Figure 1 AI-enabled SOC analytics pipeline (telemetry → feature engineering → detection → triage → response).



Evolution of Cyber Threats

Traditional malware and intrusion techniques were historically characterized by static artifacts, deterministic indicators of compromise, and repeatable signatures that enabled reliable detection through rule-based and signature-driven security mechanisms. Contemporary adversarial capabilities, however, have evolved toward highly adaptive operational models that emphasize automation, modular toolchains, and living-off-the-land (LotL) techniques. These approaches deliberately exploit legitimate system binaries, administrative utilities, and trusted execution paths in order to evade static detection and blend into benign operational baselines.

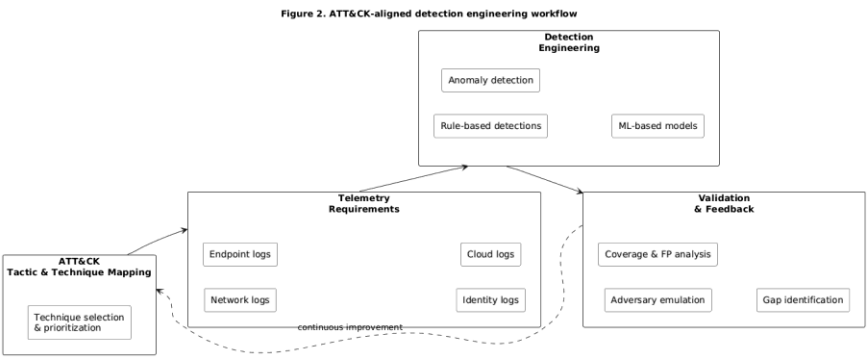
Modern multi-stage attack campaigns typically exhibit a structured progression across distinct operational phases, including initial access, credential access, lateral movement, privilege escalation, persistence, command-and-control establishment, and data exfiltration. The decomposition of these campaigns into discrete behavioral units enables systematic modeling of adversary activity. Within this context, the MITRE ATT&CK framework provides a standardized, behavior-centric taxonomy that captures adversary tactics, techniques, and procedures (TTPs) across enterprise environments, independent of specific malware families or indicators (MITRE, 2025).

ATT&CK-aligned technique mapping enables the translation of abstract threat intelligence into measurable detection objectives. By associating each technique with required telemetry sources, analytical methods, and validation criteria, detection engineering efforts can be evaluated in terms of coverage completeness, redundancy, and operational blind spots. This structured mapping further supports adversary emulation and purple-team exercises, allowing security teams to validate detection logic against realistic, technique-driven attack scenarios rather than isolated artifacts.

Ransomware ecosystems, advanced persistent threats (APT), and supply-chain attacks have emerged as dominant threat categories shaping contemporary organizational risk profiles. Empirical breach analyses indicate that ransomware operations increasingly rely on initial access brokers, affiliate-based delivery models, and multi-extortion strategies, amplifying both likelihood and potential impact (Verizon, 2024). APT campaigns are characterized by prolonged dwell times, stealthy persistence mechanisms, and tailored toolsets, significantly increasing detection complexity and downstream consequences (ENISA, 2024).

Supply-chain attacks introduce systemic risk by targeting shared dependencies, build pipelines, software update mechanisms, and managed service providers. Such attacks propagate trust violations across multiple organizations simultaneously, undermining traditional perimeter-centric security assumptions. As a result, supply-chain risk management has become a first-class concern within modern cybersecurity risk frameworks, supported by dedicated guidance such as NIST SP 800-161, which emphasizes supplier assessment, provenance assurance, and continuous monitoring across the system lifecycle (Boyens et al., 2022).

Figure 2 ATT&CK-aligned detection engineering workflow



Note. Conceptual workflow designed by the author with reference to the MITRE ATT&CK framework (MITRE, 2025).

Figure 2 illustrates an ATT&CK-aligned detection engineering workflow structured as a conceptual and iterative process. The workflow begins with the identification of adversary behaviors at the level of MITRE ATT&CK tactics and techniques. At this stage, techniques that are most relevant to protected organizational assets are selected, and threat prioritization is performed based on anticipated risk and operational relevance.

In the second stage, telemetry requirements necessary for the detection of each ATT&CK technique are defined. These telemetry sources typically include network flow records, endpoint activity events, authentication logs, cloud audit trails, and application-level logs. The identified data sources are then mapped to appropriate analytical approaches, which may include rule-based detections, statistical anomaly detection methods, and machine learning–based models, depending on data characteristics and detection objectives.

In the final stage, the developed detection mechanisms are validated through adversary emulation, red-team, or purple-team scenarios. This validation process evaluates detection performance against realistic attack flows, assesses false positive behavior, and identifies coverage gaps across the attack lifecycle. Based on the validation outcomes, detection rules and analytical models are refined, establishing a closed-loop detection engineering process that supports continuous improvement.

The figure represents a threat-driven and measurable detection engineering approach designed by the author with reference to the MITRE ATT&CK framework (MITRE, 2025).

AI-Based Threat Detection

AI-based threat detection is widely deployed across heterogeneous security telemetry sources, including network traffic, endpoint activity, identity and access management logs, and application-level audit trails. These deployments aim to identify

malicious behavior patterns that are difficult to capture through static signatures or manually curated rules. In operational environments, such capabilities are commonly integrated into Security Information and Event Management (SIEM) pipelines and extended with machine learning–based detection modules, such as user and entity behavior analytics (UEBA), anomaly scoring engines, and supervised classification components.

Network-level telemetry typically includes flow records (e.g., NetFlow and IPFIX), DNS and HTTP metadata, and Transport Layer Security (TLS) handshake attributes. These signals provide visibility into communication patterns, protocol usage, and encrypted session characteristics, enabling the detection of anomalous connectivity, command-and-control behaviors, and data exfiltration indicators. Endpoint telemetry complements network data by capturing process creation events, command-line arguments, file and registry modifications, and parent–child process relationships, which are critical for identifying execution-based attack techniques and living-off-the-land activity.

Identity-related telemetry, including authentication logs, identity provider (IdP) audit events, and privilege assignment records, supports the detection of credential misuse, lateral movement, and privilege escalation behaviors. Application and cloud audit logs further extend coverage by providing contextual information related to API usage, configuration changes, and access to sensitive resources. The fusion of these diverse telemetry sources enables multi-perspective analysis of attacker behavior across the attack lifecycle.

Feature construction plays a central role in translating raw telemetry into machine-learning–ready representations. Common feature engineering strategies include frequency-based aggregates, sliding temporal windows, and statistical summaries that capture deviations from historical baselines. Graph-based representations,

such as host–process–network interaction graphs, are increasingly employed to model relational dependencies and propagation paths across system components. In addition, categorical encodings of protocol types, application identifiers, and event attributes are used to preserve semantic context while enabling scalable learning.

The effectiveness of AI-based threat detection depends not only on detection accuracy but also on robustness under evolving operational conditions. Challenges such as severe class imbalance, non-stationary data distributions, and adversarial manipulation necessitate continuous model monitoring, periodic retraining, and integration with human analyst workflows. Consequently, AI-based detection systems are most effective when embedded within adaptive security architectures that combine automated analytics with validation, feedback, and governance mechanisms.

Table 1 Telemetry–model mapping for AI-assisted detection

Telemetry Component	Example Signals	AI/ML Model	Primary Risk	Representative Mitigations
Network	Flow, DNS, TLS metadata	Anomaly detection, autoencoders	Evasion, concept drift	Adversarial training, drift detection
Endpoint	Process, registry, command-line	Supervised classifiers	Label leakage, overfitting	Label hygiene, cross-validation
Identity	Auth logs, IdP events	Graph-based inference	Poisoning, privilege abuse	Graph sanitization, role baselines
Cloud/Application	API calls, audit logs	Sequence models	Generalization errors	Scoped training, monitoring

Source. *The conceptual framework is developed with reference to the MITRE ATT&CK framework and the NIST adversarial machine learning taxonomy (MITRE, 2025; Vassilev et al., 2025).*

Table 1 presents an illustrative mapping between major telemetry components, representative signal types, commonly

applied AI/ML methods, and primary associated risks in AI-assisted threat detection systems. At the network layer, anomaly scoring techniques are frequently applied to flow-level and encrypted traffic metadata in order to identify deviations from expected communication patterns. These approaches, however, are particularly sensitive to concept drift and adversarial evasion strategies that gradually alter traffic characteristics over time.

At the endpoint layer, supervised classifiers are commonly trained on process execution, file system activity, and registry events to distinguish benign from malicious behavior. While such models can achieve high detection accuracy for known attack patterns, they introduce risks related to label leakage and overfitting, particularly when training data reflects incomplete or biased ground truth.

Identity-centric detection increasingly relies on graph inference techniques that model relationships between users, devices, roles, and authentication events. These methods support the identification of abnormal access paths and credential abuse scenarios but remain vulnerable to poisoning attacks if adversarial activity contaminates training or reference data. The table highlights that each telemetry–model pairing introduces distinct operational risks, underscoring the need for layered defenses, data validation, and governance controls as part of AI-enabled detection architectures.

The mappings presented in the table are derived from a conceptual synthesis of ATT&CK-aligned detection practices and adversarial machine learning considerations, as discussed in (MITRE, 2025) and (Vassilev et al., 2025).

Datasets and Evaluation Considerations

Dataset-driven evaluation remains a common practice in the assessment of intrusion detection and AI-based threat detection systems, with benchmark corpora such as CIC-IDS2017 and UNSW-

NB15 being among the most frequently adopted datasets. CIC-IDS2017 provides labeled benign and malicious traffic generated under a controlled experimental setup, accompanied by a documented data generation methodology and detailed attack scenarios (Sharafaldin et al., 2018). This dataset enables reproducible experimentation and comparative evaluation across detection approaches, particularly for network-based intrusion detection tasks.

The UNSW-NB15 dataset complements earlier benchmarks by incorporating contemporary normal traffic patterns blended with synthetically generated modern attack activities, reflecting a broader range of protocol usage and attack behaviors (Moustafa & Slay, 2015). Its feature-rich design and balanced representation of multiple attack categories have contributed to its widespread adoption in the evaluation of both classical machine learning and deep learning–based intrusion detection systems.

Despite their utility, survey literature consistently highlights structural limitations associated with benchmark datasets. These limitations include realism gaps between curated datasets and operational telemetry, severe class imbalance that distorts performance metrics, and dataset-specific biases that reduce generalizability across environments (Hozouri et al., 2025). Models optimized on a single dataset may therefore exhibit inflated detection performance while failing to transfer effectively to real-world deployments.

To mitigate these limitations, complementary evaluation practices are increasingly recommended. Temporal data splits help assess model stability under non-stationary conditions by simulating deployment over time. Cross-site or cross-dataset validation supports robustness analysis across heterogeneous environments and traffic profiles. In addition, ATT&CK technique coverage scoring has emerged as a behavior-centric evaluation approach, enabling

detection performance to be measured in terms of adversary technique coverage rather than aggregate accuracy alone. Such practices reduce overfitting to a single corpus and promote evaluation designs that better reflect operational detection objectives.

Collectively, these considerations emphasize that dataset-driven evaluation should be interpreted as one component of a broader validation strategy, supplemented by behavior-based metrics, adversary emulation, and cost-sensitive performance analysis aligned with real-world security operations.

Explainability and Analyst Trust

Security operations depend on interpretable alert rationale to support efficient incident triage, prioritization, and response actions. In operational Security Operations Center (SOC) environments, opaque detection outputs hinder analyst decision-making, increase investigation time, and contribute to alert fatigue. Explainable artificial intelligence (XAI) techniques address these challenges by providing structured insight into model behavior through mechanisms such as feature attribution, rule extraction, and transparent decision summaries.

Feature attribution methods enable analysts to identify which input signals or event attributes most strongly influenced a detection outcome, supporting rapid validation and contextual understanding. Rule extraction techniques translate complex model decisions into human-readable logic, facilitating consistency checks and policy alignment. Decision summaries further enhance interpretability by aggregating model confidence, contributing factors, and historical context into concise explanations suitable for operational workflows. Collectively, these capabilities improve auditability, regulatory compliance, and analyst trust while reducing unnecessary investigative overhead.

Systematic reviews of XAI-based intrusion detection systems emphasize that explainability introduces both operational benefits and new security considerations (Khan et al., 2025). While explanatory outputs improve transparency, they may also expose sensitive model characteristics or decision boundaries that can be exploited by adaptive adversaries. Adversarial manipulation of explanations, including targeted perturbations designed to produce misleading or incomplete rationales, represents an emerging attack surface in AI-enabled security systems.

Consequently, explainability evaluation requires dedicated assessment criteria beyond conventional detection accuracy. Fidelity analysis is necessary to ensure that explanations accurately reflect underlying model behavior rather than post hoc approximations. Stability analysis under benign input perturbations supports robustness assessment by verifying that explanations remain consistent for semantically equivalent events. In addition, threat modeling for adversarial XAI scenarios is required to evaluate the resilience of explanatory mechanisms against manipulation, inference, and information leakage. These considerations position explainability not only as a usability requirement but also as a security-relevant component of AI-based threat detection architectures.

Adversarial ML and Robustness

Machine learning–driven threat detection introduces additional exposure to adversarial attacks that specifically target the learning and inference processes of detection systems. These attacks extend beyond conventional evasion techniques and exploit structural properties of machine learning models and data pipelines. Prominent adversarial threat classes include inference-time evasion, training-time poisoning, and model extraction or inference attacks,

each of which compromises detection reliability through distinct mechanisms.

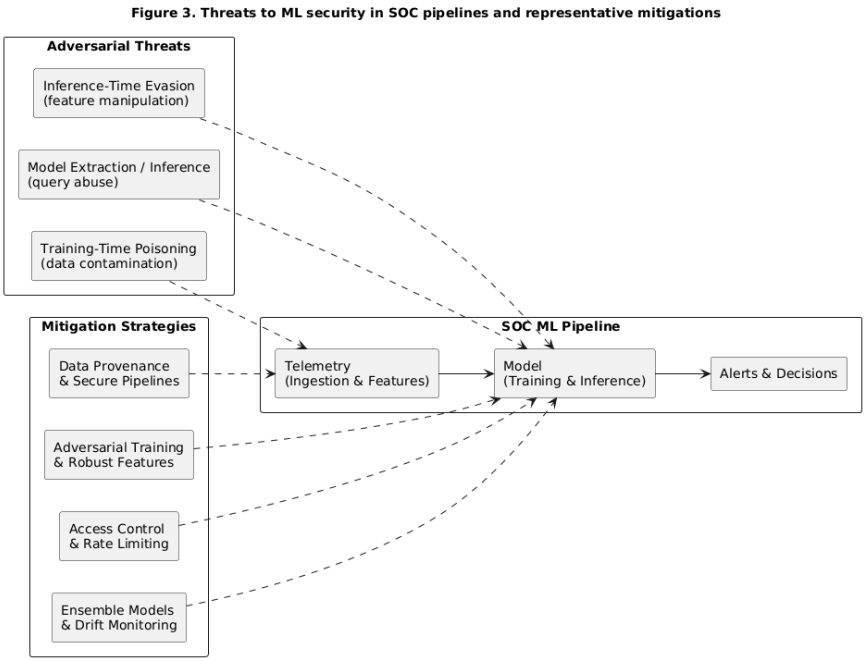
Inference-time evasion attacks manipulate input features at prediction time to induce misclassification while preserving functional equivalence from an operational perspective. Such attacks are particularly effective against anomaly-based detectors and statistical models that rely on learned decision boundaries. Training-time poisoning attacks corrupt the training dataset or reference baselines, causing systematic bias, reduced sensitivity, or targeted blind spots in deployed models. Model extraction and inference attacks aim to reconstruct model parameters or infer sensitive properties of training data through repeated queries, potentially enabling tailored evasion strategies or privacy violations.

The National Institute of Standards and Technology (NIST) provides a structured taxonomy and standardized terminology for adversarial machine learning (AML), encompassing attacker goals, knowledge assumptions, capabilities, and attack lifecycle stages (Vassilev et al., 2025). This taxonomy supports consistent threat modeling, comparative evaluation, and systematic design of defensive controls across AI-enabled security systems.

Robust detection architectures therefore require layered defense strategies that address adversarial risk throughout the model lifecycle. Adversarial training techniques improve resilience by exposing models to perturbed samples during training. Telemetry provenance controls and secure data pipelines reduce susceptibility to poisoning by ensuring data integrity and traceability. Outlier filtering mechanisms mitigate the impact of anomalous or malicious inputs, while ensemble modeling reduces single-model failure modes by diversifying detection logic. Continuous drift monitoring further supports operational robustness by identifying deviations in data distributions and model behavior over time.

Operational resilience is strengthened through adversary emulation and ATT&CK-aligned evaluation methodologies. By simulating realistic attacker behaviors and techniques, detection systems can be tested against adaptive threats in controlled environments. Such evaluations enable verification of detection coverage, identification of failure modes, and prioritization of mitigation efforts based on observed weaknesses rather than theoretical assumptions (MITRE, 2025). Together, these practices position adversarial robustness as a foundational requirement for trustworthy AI-based threat detection.

Figure 3 Threats to ML security in SOC pipelines (evasion, poisoning, extraction) and representative mitigations.



Source. *Conceptual diagram designed by the author with reference to the NIST adversarial machine learning taxonomy (Vassilev et al., 2025).*

Figure 3 illustrates the principal threats to machine learning security within Security Operations Center (SOC) pipelines and

maps these threats to representative mitigation strategies. The figure categorizes adversarial risks into three primary classes: inference-time evasion, training-time poisoning, and model extraction or inference attacks. Each threat category is positioned along the end-to-end SOC analytics pipeline, spanning data ingestion, feature engineering, model training, and deployment.

Inference-time evasion is depicted as targeting the prediction phase by subtly manipulating input telemetry to bypass learned decision boundaries. Training-time poisoning is shown as affecting earlier pipeline stages by contaminating training data, baseline profiles, or feedback loops, thereby degrading model reliability over time. Model extraction and inference threats are illustrated as exploiting deployed interfaces to recover model behavior or sensitive training characteristics.

Corresponding mitigation strategies are aligned with each threat category, including adversarial training, secure data ingestion and provenance verification, ensemble-based detection, anomaly filtering, and continuous drift monitoring. The figure represents a defense-in-depth perspective, emphasizing that adversarial resilience must be addressed across the full model lifecycle rather than through isolated countermeasures.

The conceptual workflow and threat taxonomy depicted in the figure are designed by the author with reference to the NIST adversarial machine learning taxonomy and terminology, providing a structured and standards-aligned representation of adversarial risk in AI-enabled SOC environments (Vassilev et al., 2025).

Risk Management and Decision Support Mechanisms

AI-enabled detection achieves maximum operational value when embedded within a disciplined and systematic risk management lifecycle. Effective risk management establishes a structured linkage between technical security signals—such as

alerts, anomalies, and vulnerability findings—and higher-level business considerations, including potential impact, risk treatment selection, and continuous performance monitoring. Without this linkage, detection outputs remain isolated technical artifacts with limited influence on strategic decision-making.

The NIST Risk Management Framework (RMF) provides a comprehensive and process-oriented structure for integrating security controls into organizational risk governance. The framework defines sequential and iterative steps encompassing system categorization, control selection, control implementation, control assessment, authorization to operate, and continuous monitoring (Joint Task Force, 2018). Within this structure, AI-enabled detection capabilities contribute evidence at multiple stages, including control effectiveness assessment, ongoing monitoring, and risk posture updates based on observed threat activity.

In parallel, ISO/IEC 27005 offers guidance specifically focused on information security risk management in support of the Information Security Management System (ISMS) lifecycle. The standard emphasizes systematic risk identification, analysis, and evaluation processes, followed by risk treatment, communication, and ongoing review (ISO, 2022). AI-driven detection outputs can inform each of these phases by providing quantitative and behavior-based indicators of threat likelihood, control performance, and residual risk.

When aligned with established frameworks such as NIST RMF and ISO/IEC 27005, AI-enabled detection systems support more consistent risk prioritization, evidence-based treatment decisions, and adaptive monitoring strategies. This alignment ensures that advanced detection capabilities are not deployed as isolated technical enhancements, but rather as integral components of organizational risk governance and decision support mechanisms.

Quantitative and Semi-Quantitative Risk Modeling

Risk estimation in cybersecurity commonly follows a likelihood–impact decomposition, augmented by operational indicators derived from technical, organizational, and environmental factors. This decomposition enables structured comparison of heterogeneous risk scenarios and supports prioritization under constrained resources. Threat likelihood estimation is typically informed by observed threat activity, asset exposure levels, attack surface characteristics, and the maturity of implemented security controls. Empirical telemetry, threat intelligence feeds, and historical incident data contribute to dynamic updates of likelihood assessments.

Impact estimation is derived from the potential consequences of successful compromise across confidentiality, integrity, and availability (CIA) objectives, as well as regulatory obligations, contractual requirements, and service criticality. Financial loss, operational disruption, reputational damage, and compliance penalties are commonly considered impact dimensions. Semi-quantitative scoring schemes are often employed to normalize these factors when precise monetary estimation is infeasible, enabling consistent aggregation across risk categories.

Control effectiveness represents a critical moderating factor in risk estimation and is derived from continuous monitoring outcomes, security assessments, audit findings, and post-incident analyses. Metrics such as detection coverage, response latency, false positive burden, and historical containment success provide evidence-based indicators of control performance. Incorporating control effectiveness into risk models supports estimation of residual risk rather than theoretical exposure.

AI-enabled analytics enhance both quantitative and semi-quantitative risk modeling by providing probabilistic signals and

uncertainty-aware estimates. Machine learning models can generate posterior likelihood estimates of compromise, forecast expected loss under varying threat conditions, and support scenario-driven what-if analysis. Such capabilities enable exploration of alternative mitigation strategies and assessment of trade-offs between prevention, detection, and response investments.

Cost-sensitive decision support mechanisms further refine prioritization by integrating multiple criteria into composite risk scores. These criteria commonly include detection confidence, exploitability, remediation cost, time-to-mitigate, and expected business impact. Multi-criteria decision analysis techniques allow security leaders to balance technical risk reduction against economic and operational constraints, supporting transparent and defensible risk treatment decisions. Within this context, quantitative and semi-quantitative risk models serve as a critical bridge between AI-driven detection outputs and strategic cybersecurity governance.

Governance for AI Components

Risk governance must explicitly address risks originating from artificial intelligence components in addition to conventional cybersecurity threats. Such risks include algorithmic bias, model brittleness under distributional shift, unintended misuse, and uncontrolled generalization beyond validated operating conditions. In security-critical environments, these factors may degrade detection reliability, introduce systematic blind spots, or amplify the effectiveness of adversarial manipulation, thereby altering the overall organizational risk posture.

The NIST Artificial Intelligence Risk Management Framework (AI RMF 1.0) provides lifecycle-oriented guidance for managing AI-related risks, emphasizing governance functions, measurement activities, and trustworthiness characteristics such as validity, reliability, robustness, transparency, and accountability

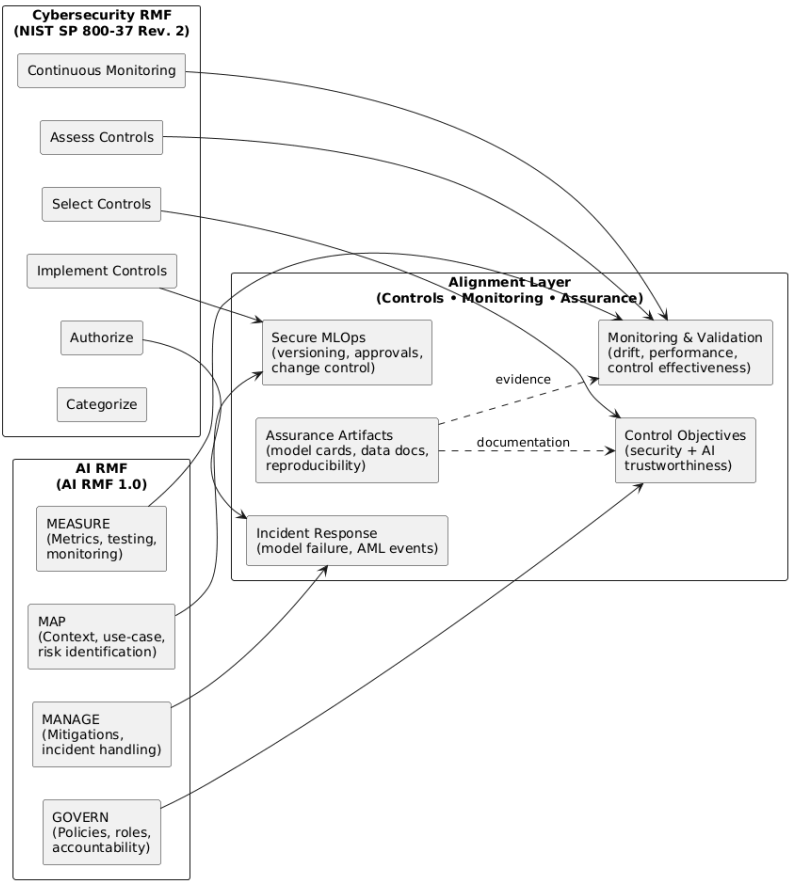
(NIST, 2023). The framework promotes structured identification, assessment, and mitigation of AI risks across design, development, deployment, operation, and retirement phases, enabling consistent oversight throughout the AI system lifecycle.

AI governance intersects directly with cybersecurity governance through shared control objectives and operational processes. Model monitoring and validation activities align with continuous security monitoring requirements, while secure MLOps practices support integrity, traceability, and controlled deployment of AI components. Change control mechanisms ensure that model updates, retraining cycles, feature modifications, and configuration changes are authorized, tested, and documented prior to operational use. Incident response procedures must therefore encompass not only infrastructure and software failures, but also model degradation, adversarial manipulation, and unexpected behavior resulting from data drift or malicious inputs.

Assurance activities provide verifiable evidence of trustworthy AI operation and support accountability, auditability, and regulatory compliance. Such activities may include the use of model cards to document intended use, performance boundaries, and known limitations; data documentation to capture dataset provenance, representativeness, and quality characteristics; reproducibility artifacts to enable independent verification of experimental results; and periodic robustness testing aligned with adversarial machine learning (AML) taxonomies (Vassilev et al., 2025; NIST, 2023). Collectively, these governance mechanisms ensure that AI-enabled detection systems remain aligned with organizational risk tolerance, security objectives, and evolving threat landscapes.

Figure 4 Integrated governance model: cybersecurity RMF × AI RMF alignment (controls, monitoring, assurance).

Figure 4. Integrated governance model: cybersecurity RMF × AI RMF alignment (controls, monitoring, assurance)



Source. *Conceptual diagram designed by the author with reference to NIST RMF and NIST AI RMF (Joint Task Force, 2018; NIST, 2023).*

Figure 4 illustrates an integrated governance model that aligns cybersecurity risk management practices with artificial intelligence–specific risk management processes. The figure conceptually maps the NIST Risk Management Framework (RMF) to the NIST AI Risk Management Framework (AI RMF), highlighting corresponding governance, control, monitoring, and assurance functions across both domains.

Within the model, cybersecurity RMF processes such as system categorization, control selection, control assessment, authorization, and continuous monitoring are aligned with AI RMF lifecycle stages, including governance, mapping, measurement, and risk management. This alignment demonstrates how AI-related risks can be incorporated into existing cybersecurity governance structures rather than managed in isolation.

The figure further emphasizes shared operational mechanisms, including continuous monitoring of controls and models, validation and verification activities, and assurance practices that provide evidence of effectiveness. By integrating AI governance with established cybersecurity risk frameworks, the model supports consistent oversight, coordinated incident response, and traceable accountability for both conventional security controls and AI-enabled detection components.

The conceptual structure depicted in the figure is designed by the author with reference to the NIST RMF and NIST AI RMF frameworks, providing a standards-aligned and unified perspective on governance for AI-enabled cybersecurity systems (Joint Task Force, 2018; NIST, 2023).

Future Perspectives and Open Research Areas

Several research directions remain central to the advancement of AI-assisted cybersecurity and continue to shape future detection and defense architectures. One prominent area concerns operational robustness under distribution shift, where detection models must maintain stable performance despite evolving network behaviors, workload changes, and adaptive adversarial strategies. Addressing distribution shift requires mechanisms for continuous learning, reliable drift detection, and controlled model adaptation that avoid catastrophic degradation while preserving detection sensitivity.

Secure MLOps for Security Operations Center (SOC) pipelines represents another critical research domain. End-to-end integrity of data ingestion, feature engineering, model training, deployment, and update processes is essential to mitigate risks associated with training-time poisoning, unauthorized model modification, and supply-chain compromise. Secure telemetry provenance, artifact signing, and controlled deployment workflows contribute to trustworthy operation and are reinforced by cybersecurity supply-chain risk management guidance provided by NIST (Boyens et al., 2022).

Standardized evaluation methodologies aligned with MITRE ATT&CK technique coverage are increasingly recognized as necessary complements to dataset-centric performance metrics. Evaluating detection effectiveness in terms of adversary technique coverage, detection latency, and failure modes supports behavior-driven assessment and reduces over-reliance on aggregate accuracy measures. Such approaches enable more meaningful comparison across detection systems and facilitate alignment between detection engineering and adversary emulation activities.

Privacy-preserving analytics under regulatory and organizational constraints constitute an additional open research challenge. Techniques such as secure aggregation, federated learning, and controlled data minimization aim to balance detection capability with confidentiality and compliance requirements. Ensuring verifiable security properties while maintaining analytical effectiveness remains an active area of investigation, particularly in multi-tenant and cross-organizational contexts.

Collectively, these directions indicate a strategic shift toward hybrid security architectures that combine classical security controls with AI-driven analytics. Such architectures are supported by formal risk management frameworks, adversarially robust model design principles, and integrated governance mechanisms that address both

cybersecurity and AI-specific risks (ENISA, 2024; Vassilev et al., 2025; Joint Task Force, 2018).

References

Boyens, J., Smith, A., Bartol, N., Winkler, K., Holbrook, A., & Fallon, M. (2024). Cybersecurity supply chain risk management practices for systems and organizations (NIST SP 800-161r1-upd1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-161r1-upd1

European Union Agency for Cybersecurity. (2024). ENISA threat landscape 2024. ENISA. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

Hozouri, A., Mirzaei, A., & Effatparvar, M. (2025). A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges. Discover Artificial Intelligence, 5, Article 314. https://doi.org/10.1007/s44163-025-00578-1

International Organization for Standardization. (2022). ISO/IEC 27005:2022 information security, cybersecurity and privacy protection—Guidance on managing information security risks. https://www.iso.org/standard/80585.html

Joint Task Force. (2018). Risk management framework for information systems and organizations: A system life cycle approach for security and privacy (NIST SP 800-37 Rev. 2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-37r2

Khan, N., Ahmad, K., Al Tamimi, A., Alani, M. M., Bermak, A., & Khalil, I. (2025). Explainable AI-based intrusion detection

systems for Industry 5.0 and adversarial XAI: A systematic review. Information, 16(12), Article 1036. https://doi.org/10.3390/info16121036

MITRE. (2025). Enterprise matrix. MITRE ATT&CK. Retrieved December 16, 2025, from https://attack.mitre.org/matrices/enterprise/

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference (MilCIS) (pp. 1–6). IEEE. https://doi.org/10.1109/MilCIS.2015.7348942

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP) (pp. 108–116). SCITEPRESS. https://doi.org/10.5220/0006639801080116

Tabassi, E. (2023). Artificial intelligence risk management framework (AI RMF 1.0) (NIST AI 100-1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

Verizon. (2024). 2024 data breach investigations report. https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf

Vassilev, A., Oprea, A., Fordyce, A., Anderson, H., Davies, X., & Hamin, M. (2025). Adversarial machine learning: A taxonomy and terminology of attacks and mitigations (NIST AI 100-2e2025). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-2e2025

Dempsey, K. L., Eavy, P., & Moore, G. (2011). Information security continuous monitoring (ISCM) for federal information systems and organizations (NIST SP 800-137). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-137

Joint Task Force. (2012). Guide for conducting risk assessments (NIST SP 800-30 Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-30r1

Joint Task Force. (2020). Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-53r5

Nelson, A., et al. (2025). Incident response recommendations and considerations for cybersecurity risk management (NIST SP 800-61 Rev. 3). National Institute of Standards and Technology. https://csrc.nist.gov/pubs/sp/800/61/r3/final

BÖLÜM 9

COMPARATIVE ANALYSIS OF TRANSFER LEARNING-BASED U-NET ARCHITECTURES FOR STRUCTURAL CRACK DETECTION

EYYÜP YILDIZ[1]

Abstract

The temporal changes in the structural forms of buildings used in daily life are observed. Detecting cracks in high-cost and critically important structures such as natural gas pipelines, underwater internet cables, and dam walls is crucial for taking necessary precautions. Automatic detection of cracks, especially in underwater structures, allows for early preventative measures. In this context, deep learning models, which are increasingly used in all fields, have become frequently employed in the segmentation of cracks found in

[1] Dr. Öğr. Üyesi, Erzincan Binali Yıldırım Üniversitesi, Bilgisayar Mühendisliği, Orcid: 0000-0002-7051-3368

underwater images. This study investigates the segmentation performance of different convolutional structures trained on a dataset of underwater images. For this purpose, the U-Net framework, which has a groundbreaking impact on image segmentation problem solving, was utilized. To improve generalization under limited training data and adverse visual conditions, ImageNet-pretrained encoder backbones are integrated into U-Net. Specifically, U-Net variants with ResNet34, DenseNet121, and EfficientNet-B3 encoders are trained under the same preprocessing and training process. Dice loss is employed to mitigate severe class imbalance between crack and background pixels. Model performance is evaluated using F1-score, Intersection over Union (IoU), and recall. Results indicate that the EfficientNet-B3-based U-Net achieves the best overall performance. These findings highlight that appropriate backbone selection significantly influences both the detection of fine cracks and the overall segmentation quality in underwater scenarios.

Introduction

Nowadays, Deep learning is applied various fields, such as route planning (Aslan Yıldız et al., 2025; Taş et al., 2025), image generation (Gou et al., 2025; J. Lin et al., 2020). Many structures built today utilize concrete as building material. Concrete is heavily used in underwater structures due to its positive properties such as durability and longevity (Nguyen et al., 2023; Zheng et al., 2025). However, cracks can develop in underwater structures over time due to internal factors such as load distribution and material composition,

as well as external factors such as chemical interaction or collision with the external environment (Ai et al., 2023; Y. Li et al., 2022; Orinaitė et al., 2023). Maintenance, renovation, and construction of underwater structures are considerably more costly than structures located above water. Therefore, the timely and effective detection of cracks in underwater structures is a crucial issue.

In practice, the demand for automatic underwater structural crack detection has increased rapidly every year. While imaging underwater structures is quite challenging, the resulting images present numerous problems due to the nature of water, such as low resolution, turbidity, and distorted shapes due to light refraction (Ali et al., 2022; Y. Li et al., 2022; Zhu et al., 2024). Therefore, crack detection from underwater images is even more difficult than from surface images. Among the methods proposed to solve this problem, convolutional neural networks (CNNs) stand out. The spatial, neighborhood-based operation of the convolution process on the image makes CNNs successful in crack segmentation from images of underwater structures(Ai et al., 2023).

Originally introduced for biomedical image segmentation, U-Net (Ronneberger et al., 2015) has become a valuable framework architecture for pixel-level segmentation. symmetric encoder–decoder design and skip connections of U-Net fuse high-resolution encoder features with decoder features. Consequently U-Net is able to detect boundaries and details of input images (Ai et al., 2023). In recent years, U-Net has been widely used in underwater crack segmentation where accurate pixel-level delimitation is of paramount importance. Despite its strong performance, training U-Net based CNNs typically require large, densely annotated datasets, whereas underwater crack segmentation datasets (e.g., the Ren dataset [4]) are relatively limited in scale compared with large-scale image datasets such as ImageNet (Deng et al., 2010) and MS-COCO (T.-Y. Lin et al., 2015) . As a result, models trained solely on small,

task-specific underwater datasets may not fully realize their capacity and can suffer from reduced generalization under challenging underwater conditions. (Zeiler & Fergus, 2014) shows that CNNs learn generic low-level features (e.g., edge/texture/color patterns) in early layers that transfer across tasks. On the other hand, deeper layers become increasingly task specific. This observation motivates transfer learning and fine-tuning strategies, particularly when labeled data are scarce.

This study systematically investigates the effect of different encoder backbones under the U-Net framework for pixel-level (semantic) segmentation of underwater crack images. For this purpose, three different encoders, initialized with pre-trained weights on ImageNet and integrated with the U-Net architecture, were selected: ResNet34 (He et al., 2015), DenseNet121 (Huang et al., 2017), and EfficientNet-B3 (Tan & Le, 2019). Pre-trained encoders on the ImageNet dataset were used to accelerate learning and increase generalization power under limited label data conditions. These three selected U-Net-based models were fine-tuned on the (Teng, 2025) dataset. Thus, these architectures were made usable for automatic segmentation of cracks in underwater images. Experimental results showed that all three architectures have high performance potential in the underwater crack segmentation problem in terms of F1-score, IoU, and Recall metrics.

In this study, we propose three U-Net based segmentation frameworks for crack detection in underwater structure images. The main contributions of this work are as follows:

1. We conduct a comparative analysis of three ImageNet-pretrained encoder backbones (ResNet34, DenseNet121, and EfficientNet-B3) integrated into the U-Net framework. This comparison is particularly important for revealing the trade-off between segmentation accuracy and computational cost.

2. We employ the Dice loss function instead of standard cross-entropy during training to mitigate the severe class imbalance.

Literature Review

Crack detection has become major challenge of computer vision research. Studies in literature can generally be categorized into two main groups: traditional image processing methods based on hand-craft features and deep learning-based approaches that learn from data.

The first methods proposed for the crack segmentation problem are based on threshold values. These methods focus on determining whether the pixel density values in the images exceed the threshold value being sought. The goal is to separate crack-containing image patches from background images. Li and Liu (2008) (Q. Li & Liu, 2008)proposed a pavement image-thresholding algorithm based on the neighboring difference histogram method. Fujita and Hamamoto (2011) (Fujita & Hamamoto, 2011) proposed a novel method consisting of two preprocessing and two detection steps. In the preprocessing steps, first the median filter is applied to image and second multi-scale line filter with the Hessian matrix is used to emphasize cracks against blebs or stains. In the detection steps dynamic thresholding approach and probabilistic relaxation technique were utilized to detect the cracks. Talab et al. (2016) (Talab et al., 2016) have employed three stage methods to detect cracks in images. They first utilized Sobel filter to gray scale image,

second uses a suitable threshold to divide image into two groups: background and foreground. Lastly, they utilized Otsu method to detect major cracks inside images. Generally, threshold-based crack detection approaches can achieve fast and accurate detection on ideal images that contain cracks with smooth intensity of the surroundings. However, this is impractical for real-world scenarios where there could be various noisy pixels due to background textures and lighting conditions during the data acquisition process.

Classical machine learning methods yield more successful results than threshold-based methods in crack segmentation problems. Li et al. (2017) (G. Li et al., 2017) proposed support vector machine-based method for extracting cracks from concrete bridge images. First, they build a high-precision image acquisition framework and then applied the linear support vector machine using greedy search strategy for noise elimination to image. Shi et al. (2016) (Shi et al., 2016) proposed a pavement crack detection framework namely CrackForest to identify complex cracks. CrackForest applies the integral channel features to redefine the tokens that constitute a crack and introduce random structured forests to generate a high-performance crack detector. This method is shown to be able to extract complex cracks with arbitrary topology (e.g., corners, curves, lines).

The superiority of CNN architectures in learning local and global pixel neighborhoods, and their problem-specific structures, make CNN architectures stand out compared to other artificial intelligence models in the image segmentation problem. Many crack image processes problems such as behavior analysis of crack through time require accurate detection at pixel-level of cracks (Liu et al., 2019). Long et al. (2015) (Long et al.) proposed Fully Convolutional Networks based on the traditional CNN frameworks. Compared with traditional methods, Fully Convolutional Networks are abelt to get different size of input images and significantly accelerates the

processing speed. Huyan et al. (2020) (Huyan et al., 2020) proposed a deep learning model based on the UNet, namely CrackUnet, for pixel-wise pavement crack detection. CrackUnet does not require pre-trained CNNs or data argumentation. CrackUnet utilizes constant kernel size to learn high-level features from input images. During the training, CrackUnet learns and aggregates multi-scale and high-level features from the early convolutional layers to the high-level convolutional layers, which is different from the standard approaches of only using the last convolutional layer. CrackUnet provides integrated direct supervision for features of each convolutional stage. CrackUnet apply both guided filtering and Conditional Random Fields methods to refine the final prediction results.

Materials and Methods

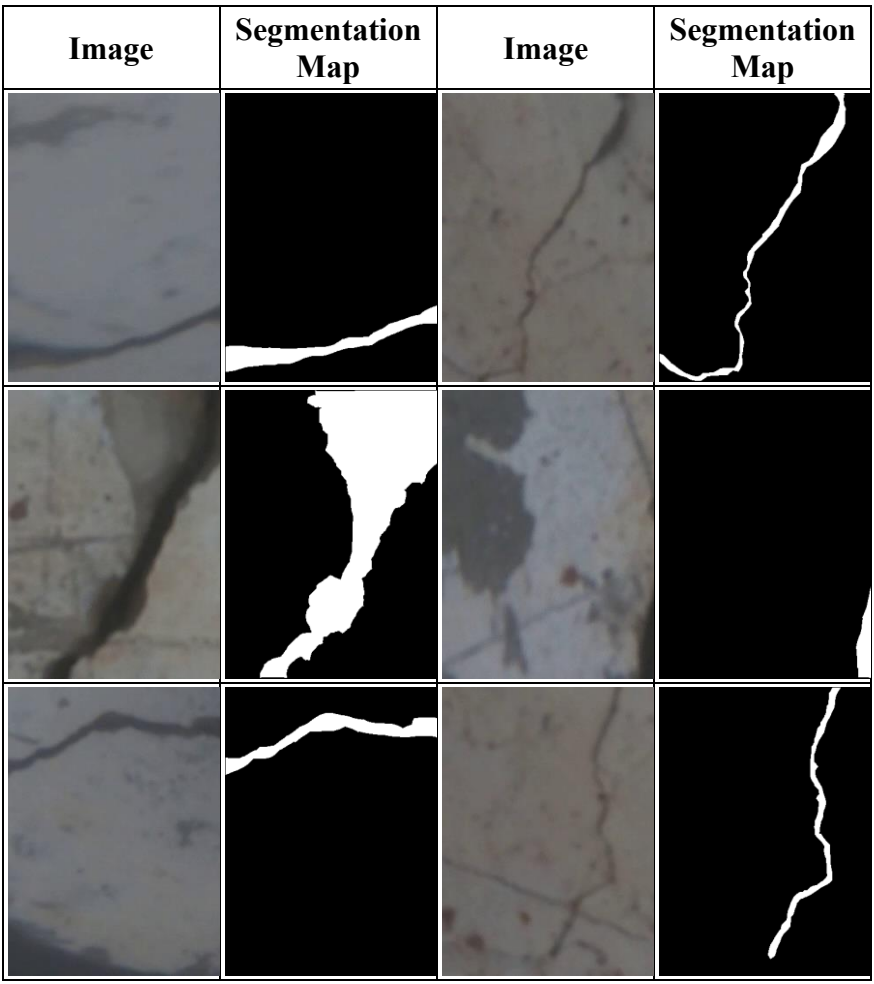
This section details the methodology of the proposed deep learning-based underwater crack segmentation framework.

Dataset

This section details the methodology of the proposed deep learning-based underwater crack segmentation framework. In this study, we used the publicly available “Underwater Crack Detection” dataset (Teng, 2025). The dataset consists of high-resolution underwater images, each accompanied by pixel-level binary mask as shown in Figure 1. In these masks, crack pixels are represented in white, whereas background pixels are represented in black. To objectively assess the generalization capability of the proposed models, we

randomly partitioned the dataset into three subsets: 80% training, 10% validation, and 10% test subsets.

Figure 1. Sample images from dataset used in this study.



Preprocessing and Augmentation

Three preprocessing steps were applied. First, all input images and their corresponding masks were resized to 320 × 320 pixels. Second, to improve the effectiveness of the transfer-learning strategy, the input images were normalized using the ImageNet statistics with

mean $\mu = [0.485, 0.456, 0.406]$ and standard deviation $\sigma = [0.229, 0.224, 0.225]$. Third, horizontal/vertical flips, random rotations, and perspective transformations were applied with a probability of 50% during training to reduce memorization and improve robustness. These augmentations encourage the model to learn crack-related features that are less sensitive to orientation and viewpoint changes.

Model Architecture: Transfer Learning Based U-Net

In this study, we utilized the U-Net architecture as the backbone for the semantic segmentation of (Teng, 2025) dataset. U-Net consists of an encoder that extracts contextual information from the input image and a decoder that processes this information to segmentation. The skip connections between the encoder and decoder blocks enable the reusing of spatial details. Therefore U-Net detects boundaries of segmentation and fine-structure preservation.

Instead of using the standard U-Net encoder, we integrate three ImageNet-pretrained backbones to enhance feature extraction: ResNet34, DenseNet121, and EfficientNet-B3. ResNet34 addresses the vanishing-gradient problem commonly observed in deep networks through residual blocks and shortcut connections. DenseNet121 employs dense connectivity, where each layer receives inputs from all preceding layers. This structure maximizes feature reuse and is particularly effective in preserving subtle patterns. EfficientNet-B3 is based on compound scaling, which jointly and proportionally scales network depth, width, and input resolution using a single scaling coefficient.

Loss Function and Performance Metrics

Semantic segmentation of crack images is inherently an imbalanced learning problem, due to majority of pixels in an image belonging to the background class. Under such conditions, the standard cross-

entropy loss may yield a relatively low error even when the model predominantly predicts background. To address this issue, we employ the Dice Loss function. Dice Loss maximizes the overlap between the predicted mask $P_i$and the ground-truth mask G_i, as defined in Eq. (1). Here, ε is a small constant introduced to prevent division zero. Since Dice loss emphasizes region overlap rather than raw pixel counts, it typically yields more stable optimization for detecting thin crack patterns.

$$L_{Dice} = 1 - \frac{2 \times \sum_i (P_i \times G_i) + \varepsilon}{\sum_i (P_i) + \sum_i (G_i) + \varepsilon} \quad (1)$$

We utilized IoU, F1-score and Recall metrics to assess performance of models. IoU is defined as the ratio of the intersection area to the union area between the predicted and ground-truth regions and serves as a primary indicator of segmentation quality. F1-score is the harmonic mean of precision and recall and gives a balanced measure of detection performance. Recall quantifies the proportion of true crack pixels that are correctly identified. From a structural safety perspective, achieving high recall is particularly important because missing cracks can have severe consequences in real-world inspections.

Implementation Details

All models were implemented using the. Training was performed on NVIDIA GeForge GTX 1660 Ti GPU, Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz hardware. Adam optimizer was used as with an initial learning rate of 0.0001. Models were trained for 50 epochs with a batch size of 8, and the best model weights were recorded based on the IoU score in the validation set.

Results

In this section, we evaluate in detail the training results of three U-Net-based segmentation models (Teng, 2025) whose encoder structures were pre-trained on the ImageNet dataset using ResNet34, DenseNet121, and EfficientNet-B3 architectures. Table 1 shows the Dice loss, F1 score, IoU, and Recall values of the three models trained over 50 iterations. Accordingly, the U-Net model with the EfficientNet-B3 encoder structure outperformed the other models in all performance values.These results indicate that the stronger feature extraction capability of EfficientNet-B3 enables more consistent learning of crack regions and leads to superior segmentation quality on the training data. However, the fact that Resnet34 and Densenet121-based U-Net models perform at similar levels indicates that feature reuse and feature transfer to subsequent layers in Resnet34 exhibit similar behavior on the dataset (Teng, 2025).

Tablo 1. The best result metrics of the models on the training dataset.

Model	Loss	F1 Score	IoU	Recall
Resnet34	0.2744	0.7259	0.5730	0.7259
Densenet121	0.2824	0.7180	0.5640	0.7320
EfficientNet-B3	**0.1759**	**0.8245**	**0.7045**	**0.8314**

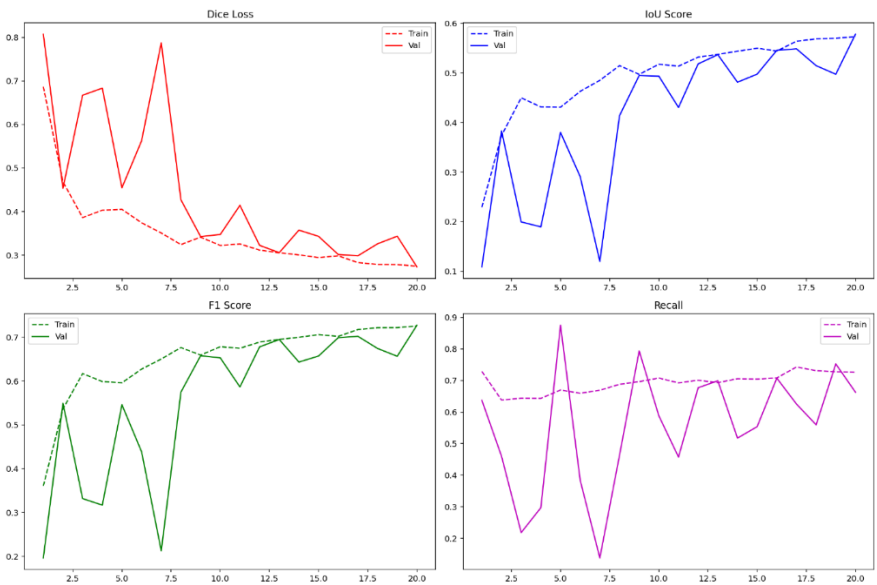
Table 2 shows the Dice loss, F1 score, IoU, and Recall values for the three models on the validation dataset. The EfficientNet-B3-based model achieves the best segmentation performance with F1 = 0.7622 and IoU = 0.6282. Overall, these findings demonstrate that backbone selection has a substantial impact on both regions overlap (IoU) and crack-capture capability (recall) in underwater crack segmentation, and that EfficientNet-B3 provides the most suitable encoder in this study due to its favorable accuracy–efficiency trade-off. Figures 1–3 illustrate the evolution of Dice loss, IoU, F1-score, and recall over

50 epochs on the training and validation sets for the U-Net models with ResNet34, DenseNet121, and EfficientNet-B3 encoder backbones.

Tablo 2. The best result metric values of the models on the training validation set.

Model	Loss	F1 Score	IoU	Recall
Resnet34	0.2728	0.7274	0.5779	0.7524
Densenet121	0.2824	0.7076	0.5640	0.7515
EfficientNet-B3	**0.2218**	**0.7622**	**0.6282**	**0.8662**

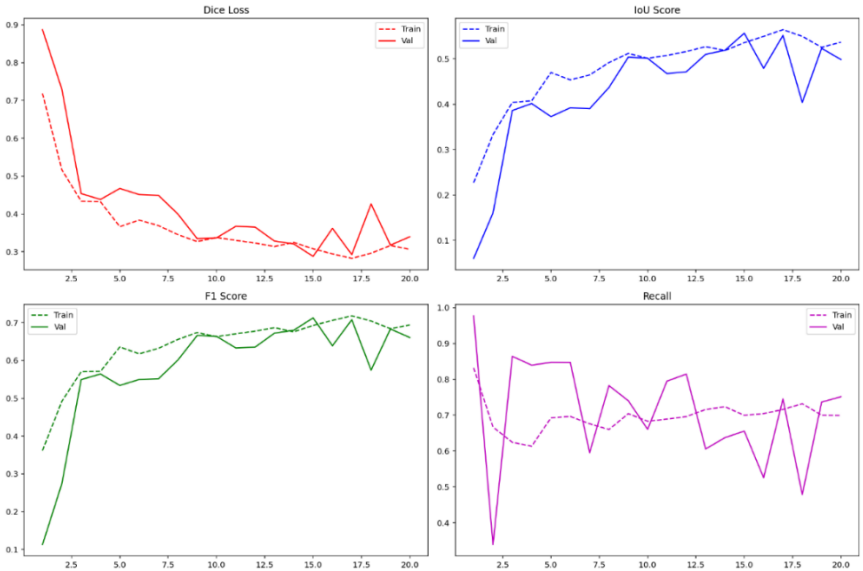
Figure 1. Training phase iteration output for ResNet34-based U-Net.



(Teng, 2025) The results we obtained from the training and validation data of the (Teng, 2025) dataset showed that the U-Net model, which used EfficientNet-B3 in its encoder structure achives better segmentation performance on all evaluation metrics. While the

ResNet34 model exhibits a more constrained yet relatively stable learning, DenseNet121 shows more pronounced fluctuations and occasional performance drops in the validation metrics. Overall, the learning curves suggest that the representational capacity of the backbone and its accuracy, efficiency balance significantly influence segmentation quality in crack segmentation, and that EfficientNet-B3 is the most suitable encoder choice within this study.

Figure 2. Training phase iteration output for U-Net based on DenseNet121.



Tablo 3. The best performance metric values of the models on the training test set.

Model	Loss	F1 Score	IoU	Recall
Resnet34	0.3945	0.6057	0.4751	0.5616
Densenet121	0.3010	0.6994	0.5715	0.8010
EfficientNet-B3	**0.2442**	**0.7560**	**0.6299**	**0.9259**

Figure 3. Training phase iteration output for EfficientNet-B3-based U-Net

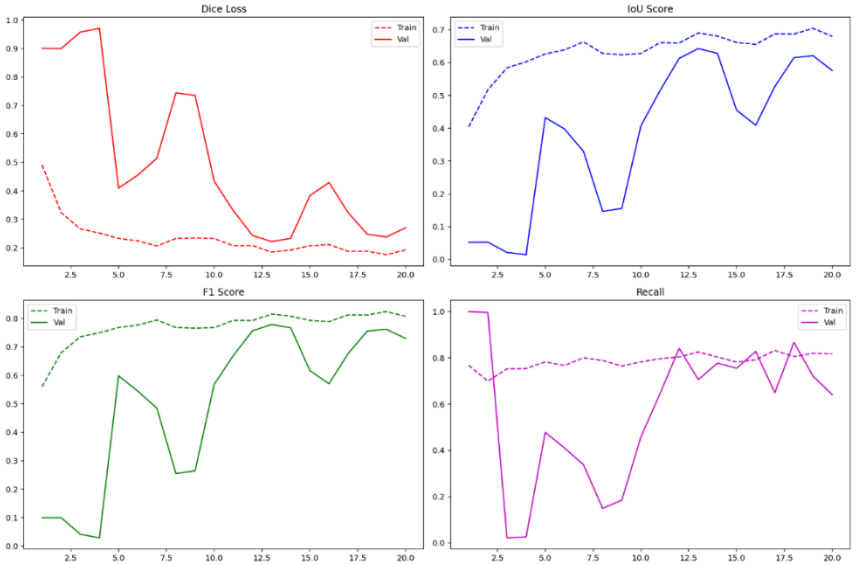


Figure 4. visual segmentation results of models.

Model	Input Image	Grand Truth	Model Result	Input Image	Grand Truth	Model Result
ResNet34						
DenseNet121						
EfficientNet-B3						

Table 3 summarizes the performance of the models on the test set. The results show that EfficientNet-B3 based U-Net achieves the best generalization among all backbones. DenseNet121 yields the second-best performance. Notably low recall of ResNet34 indicates

that a substantial portion of crack pixels might be missed. These findings confirm that the representational capacity of the encoder backbone markedly influences segmentation performance.

Figure 5 presents a qualitative comparison of the segmentation outputs produced by U-Net models with ResNet34, DenseNet121, and EfficientNet-B3 encoder backbones on two representative test images. For each backbone, the corresponding row includes the input image, the ground-truth mask, and the predicted mask. While all models can detect the main crack trajectory to some extent, noticeable differences emerge in terms of boundary accuracy, crack continuity, and the detection of fine branches. Given that small boundary deviations can substantially affect overlap-based metrics such as IoU and F1 under low-contrast and noisy conditions, these qualitative results provide important support for interpreting the quantitative findings.

As a result, the qualitative comparison indicates that the EfficientNet-B3-based U-Net produce masks better than ResNet34, DenseNet121 models. EfficientNet-B3 better preserves the continuity of crack patterns while limiting irregular mask thickness and unnecessary spillovers. In contrast, ResNet34 exhibits local thinning/thickening and noisier boundaries in some regions. DenseNet121 shows a more pronounced tendency toward over-segmentation, with visible spillovers. These observations suggest that the stronger feature-extraction capability of EfficientNet-B3 enables a more faithful representation of crack morphology, which is consistent with its higher IoU/F1 scores and its high recall in capturing crack pixels.

Conclusion

In this study, we investigated the impact of different encoder backbones-based U-Net architectures within transfer learning on underwater crack segmentation performance. The results obtained revealed that all models demonstrated above-acceptable success in the crack segmentation problem. The U-Net models we created using ResNet34 and Densenet121 showed similar levels of success on training, validation, and test data. This indicates that the ResNet34 and Densenet121 architectures closely backpropagate the fault gradients, thus converging the models to similar characteristics. On the other side, the results indicate that the EfficientNet-B3 based U-Net achieves the best overall performance in terms of F1-score, IoU, and recall metrics. The qualitative comparisons further support the EfficientNet-B3 produces masks with better crack continuity and fewer irregular spillovers, while ResNet34 tends to miss portions of thin cracks and DenseNet121 more frequently exhibits over-segmentation. In conclusion, this study demonstrated, through quantitative and qualitative results, how reliably the feature extraction capacity and representation efficiency of the encoder backbone can separate subtle, low-contrast crack structures from visually distorted underwater backgrounds.

References

Ai, D., Jiang, G., Lam, S. K., He, P., & Li, C. (2023). Computer vision framework for crack detection of civil infrastructure—A review. In *Engineering Applications of Artificial Intelligence*

(Vol. 117). Elsevier Ltd. https://doi.org/10.1016/j.engappai.2022.105478

Ali, R., Chuah, J. H., Talip, M. S. A., Mokhtar, N., & Shoaib, M. A. (2022). Structural crack detection using deep convolutional neural networks. In *Automation in Construction* (Vol. 133). Elsevier B.V. https://doi.org/10.1016/j.autcon.2021.103989

Aslan Yıldız, Ö., Sarıçiçek, İ., & Yazıcı, A. (2025). A Reinforcement Learning-Based Solution for the Capacitated Electric Vehicle Routing Problem from the Last-Mile Delivery Perspective. *Applied Sciences (Switzerland)*, *15*(3). https://doi.org/10.3390/app15031068

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, & Li Fei-Fei. (2010). ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/cvpr.2009.5206848

Fujita, Y., & Hamamoto, Y. (2011). A robust automatic crack detection method from noisy concrete surfaces. *Machine Vision and Applications*, *22*(2), 245–254. https://doi.org/10.1007/s00138-009-0244-5

Gou, Y., Li, M., Zhang, Y., Zhang, X., & He, Y. (2025). Multiple one-shot image generation via deep structure reshuffle. *Neural Networks*, *192*. https://doi.org/10.1016/j.neunet.2025.107862

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. http://image-net.org/challenges/LSVRC/2015/

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *IEEE Conference*

on Computer Vision and Pattern Recognition, 4700–4708. https://github.com/liuzhuang13/DenseNet.

Huyan, J., Li, W., Tighe, S., Xu, Z., & Zhai, J. (2020). CrackU-net: A novel deep convolutional neural network for pixelwise pavement crack detection. *Structural Control and Health Monitoring*, *27*(8). https://doi.org/10.1002/stc.2551

Li, G., Zhao, X., Du, K., Ru, F., & Zhang, Y. (2017). Recognition and evaluation of bridge cracks with modified active contour model and greedy search-based support vector machine. *Automation in Construction*, *78*, 51–61. https://doi.org/10.1016/j.autcon.2017.01.019

Li, Q., & Liu, X. (2008). Novel approach to pavement image segmentation based on neighboring difference histogram method. *Proceedings - 1st International Congress on Image and Signal Processing, CISP 2008*, *2*, 792–796. https://doi.org/10.1109/CISP.2008.13

Li, Y., Bao, T., Huang, X., Chen, H., Xu, B., Shu, X., Zhou, Y., Cao, Q., Tu, J., Wang, R., & Zhang, K. (2022). Underwater crack pixel-wise identification and quantification for dams via lightweight semantic segmentation and transfer learning. *Automation in Construction*, *144*. https://doi.org/10.1016/j.autcon.2022.104600

Lin, J., Pang, Y., Xia, Y., Chen, Z., & Luo, J. (2020). *TuiGAN: Learning Versatile Image-to-Image Translation with Two Unpaired Images*. http://arxiv.org/abs/2004.04634

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision*, 740–755. http://arxiv.org/abs/1405.0312

Liu, Y., Yao, J., Lu, X., Xie, R., & Li, L. (2019). DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, *338*, 139–153. https://doi.org/10.1016/j.neucom.2019.01.036

Long, J., Shelhamer, E., & Darrell, T. (n.d.). *Fully Convolutional Networks for Semantic Segmentation*.

Nguyen, S. D., Tran, T. S., Tran, V. P., Lee, H. J., Piran, M. J., & Le, V. P. (2023). Deep Learning-Based Crack Detection: A Survey. *International Journal of Pavement Research and Technology*, *16*(4), 943–967. https://doi.org/10.1007/s42947-022-00172-z

Orinaitė, U., Karaliūtė, V., Pal, M., & Ragulskis, M. (2023). Detecting Underwater Concrete Cracks with Machine Learning: A Clear Vision of a Murky Problem. *Applied Sciences (Switzerland)*, *13*(12). https://doi.org/10.3390/app13127335

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Nternational Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3434–3445. https://doi.org/10.1109/TITS.2016.2552248

Talab, A. M. A., Huang, Z., Xi, F., & Haiming, L. (2016). Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik*, *127*(3), 1030–1033. https://doi.org/10.1016/j.ijleo.2015.09.147

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *International Conference on Machine Learning*, 6105–6114.

Taş, M. B. H., Özkan, K., Sarıçiçek, İ., & Yazici, A. (2025). Transportation Mode Selection Using Reinforcement Learning in Simulation of Urban Mobility. *Applied Sciences (Switzerland)*, *15*(2). https://doi.org/10.3390/app15020806

Teng, S. (2025). *Underwater Crack Image Dataset*. Https://Data.Mendeley.Com/Datasets/G962rbm77b/1.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. *European Conference on Computer Vision*, 818–833. http://arxiv.org/abs/1311.2901

Zheng, L., Tan, H., Ma, C., Ding, X., & Sun, Y. (2025). A real-time crack detection approach for underwater concrete structures using sonar and deep learning. *Ocean Engineering*, *322*. https://doi.org/10.1016/j.oceaneng.2025.120582

Zhu, G., Liu, J., Fan, Z., Yuan, D., Ma, P., Wang, M., Sheng, W., & Wang, K. C. P. (2024). A lightweight encoder–decoder network for automatic pavement crack detection. *Computer-Aided Civil and Infrastructure Engineering*, *39*(12), 1743–1765. https://doi.org/10.1111/mice.13103

BÖLÜM 10
University Artificial Intelligence

A Fully AI-Generated University Learning Environment for Graduate Education in Engineering and Educational Sciences

Dr. OMER SEVINC[1]

1. Introduction

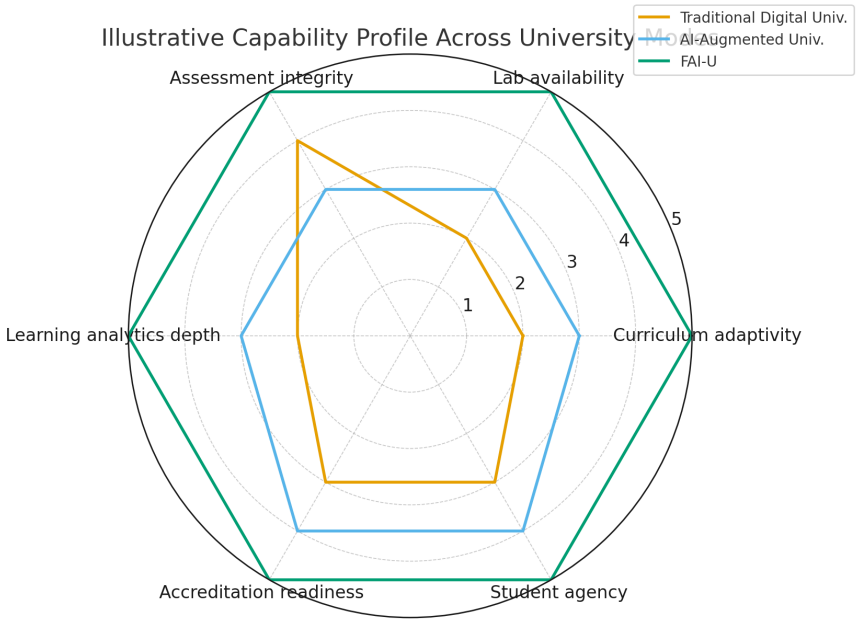
Universities are among the most complex learning institutions ever created. They coordinate knowledge production, professional credentialing, civic development, and research socialization across multiple disciplines and generations. Unlike most educational systems, universities simultaneously function as curricular systems (they define formal outcomes), epistemic systems (they decide what counts as legitimate knowledge), and social systems (they form communities that transmit norms and identities). Yet the environments through which universities enact these functions have remained structurally conservative. Even during the rapid digitization of the early 2020s, institutions often transplanted legacy pedagogy into learning management systems and videoconferencing tools rather than redesigning the university as a native digital ecology.

This conservatism partly reflects a rational institutional contract. Universities are accountable to accreditation frameworks, public legitimacy, and the slower rhythms of disciplinary change. Their authority depends on stable standards, credible assessments, and reliable credentialing. However, the consequence is that learning environments lag behind technical and pedagogical possibilities. Graduate education, in particular, remains bounded by fixed syllabi, finite lab access, supervision bottlenecks, and episodic feedback cycles. These constraints limit the frequency of deliberate practice and slow the transition from knowledge consumption to knowledge production.

In contemporary higher education practice, artificial intelligence (AI) remains mostly additive. AI tools assist with micro-tutoring, grading bounded tasks, recommending readings, or administrative triage. These applications improve efficiency but do not change the university's operating logic: faculty define outcomes and syllabi, deliver instruction, mediate discourse, and evaluate mastery through discrete assessments. The institution remains human-run with AI support, rather than AI-run under human governance.

Figure 1. *Illustrative Capability Profile Across University Models.*

[1] Dr. Omer Sevinc OMÜ Bilgisayar Programcılığı



The emergence of agentic large language models, multimodal simulators, and retrieval-grounded generation changes the feasibility frontier. AI can now generate not only content but also the conditions of learning—tasks, dialogues, laboratories, feedback loops, assessment rubrics, improvement cycles, and evidence logs—at sustained scale. This enables a structural redesign in which universities can operate continuously adaptive learning environments rather than static course containers.

This chapter therefore proposes a Fully AI-Generated University Learning Environment (FAI-U), where coordinated AI agents generate and operate the learning environment end-to-end, while humans govern aims, constraints, and legitimacy. The focus on graduate education is deliberate. Master's and PhD students require deep conceptual integration, high autonomy, and methodological rigor. Engineering graduate education depends on high-frequency laboratory practice, simulation, design iteration, and safety reasoning; AI-generated micro-labs can remove scarcity and accelerate iteration cycles. Educational sciences emphasize theory comparison, critical argumentation, and validity-driven evaluation of learning systems; they are ideal environments to theorize, test, and refine AI-native pedagogy as illustrated in Figure-1.

To clarify the institutional shift proposed here, Table 1 contrasts traditional digital universities, AI-augmented universities, and Fully AI-Generated Universities across core dimensions. The contrast frames FAI-U as a structural transformation rather than a tool-level upgrade.

Table 1. *Comparison of institutional modes.*

Dimension	**Traditional Digital University**	**AI-Augmented University**	**Fully AI-Generated University (FAI-U)**
Curriculum	Static multi-year revisions	Static core, AI supplements	Living curriculum updated continuously
Instruction	Human lectures/seminars	Human + local AI tools	AI agents generate multimodal instruction
Labs	Scarce physical labs	Some virtual labs	On-demand AI micro-labs + twin studios
Assessment	Written exams/projects	AI-assisted grading	Continuous + Socratic oral + portfolios
Analytics	End-of-term reporting	Predictive risk tools	Learner twins + intervention simulation
QA/Accreditation	Manual evidence collection	Partial automation	Live evidence stream auto-generated
Faculty role	Delivery + grading + admin	Delivery + governance	Epistemic supervision + mentorship
Student agency	Moderate, course-bounded	Higher support but tool-dependent	High; justification-centered

2. Conceptual Background and Related Work

AI in education has evolved through successive waves of representational and institutional capacity. Early rule-based tutoring systems demonstrated that individualized feedback could be scaled, but they remained brittle and expensive to author. Later waves of learning analytics and data-driven personalization made adaptivity more scalable, yet still constrained to fixed curricular shells. Generative AI introduces a deeper capability: models can produce coherent explanations, scaffold inquiry, generate examples and counterexamples, sustain open-ended tutoring, and simulate complex disciplinary laboratories across graduate-level topics.

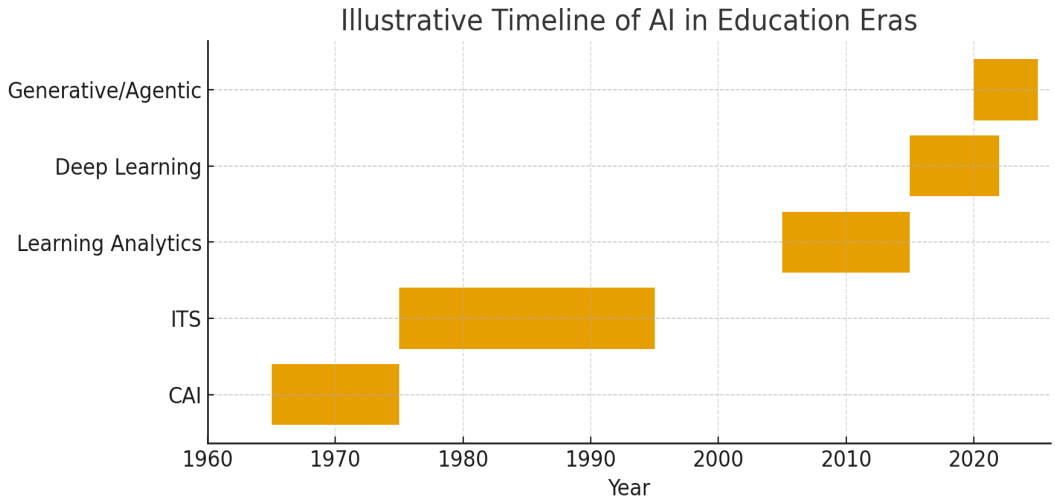
Despite this technical shift, higher-education adoption remains component-based. Universities deploy AI in local pockets—an AI grader here, a recommendation agent there—leaving the institutional environment fragmented. Graduate study is still organized around fixed syllabi, finite contact hours, and human bottlenecks that cap feedback

frequency. As a result, most institutions are drifting toward an AI-augmented university rather than an AI-native university.

Digital-twin approaches provide a critical bridge. A campus twin represents resources, spaces, laboratories, and virtual studios that can be simulated and optimized. A learner twin represents an evolving multidimensional map of mastery, pacing, self-regulation, affect, epistemic stance, and research identity. Together, these twins shift education from reactive remediation to predictive intervention by enabling forecasted risk and simulated support strategies.

Universities are also social-epistemic ecosystems. Graduate learners do not merely acquire content; they become knowledge producers by participating in scholarly communities that enact norms of evidence, critique, and contribution. Multi-agent AI architectures mirror this ecology by distributing roles across specialized agents (professor, TA, reviewer, librarian, lab manager, cohort facilitator), thereby improving pedagogical richness, auditability, and institutional alignment.

Figure 2. *Illustrative Timeline of AI in Education Eras.*



Finally, graduate education has long relied on oral defenses and dialogic assessment because advanced expertise is not reducible to discrete facts. Generative AI amplifies integrity pressures in written-product assessments, increasing the value of Socratic oral assessment that probes reasoning directly, in real time, and under cross-examination.

2.1 A Detailed History of AI in Education (AIED)

The evolution of artificial intelligence in education (AIED) is best understood as a sequence of paradigm shifts in how learning systems represent knowledge, model learners, and scale feedback. The first wave began with computer-assisted instruction (CAI) in the

1960s. Systems such as PLATO introduced interactive tutorials and early networked learning communities using scripted branching logic. While not adaptive AI, CAI established the enduring institutional ambition to scale individualized instruction.

The second wave (1970s–1990s) formed the Intelligent Tutoring Systems (ITS) era, sometimes called Intelligent Computer-Aided Instruction (ICAI). Research systems like SCHOLAR and SOPHIE used symbolic AI to represent domain knowledge and to reason about student actions. ITS architectures matured into the triad of a domain model, a student model, and a pedagogical model. Empirical studies showed substantial learning gains in structured domains but also revealed a knowledge-engineering bottleneck: authoring and maintaining expert rules was expensive, and transfer to open-ended domains was limited.

The third wave (2000s–mid-2010s) shifted toward data-driven adaptivity and learning analytics. Bayesian Knowledge Tracing, item-response modeling, and recommender systems inferred student state from large interaction logs. This period aligned with learning management systems and MOOCs, enabling access and personalization to grow but leaving university learning structurally static: adaptivity adjusted within predefined curricula and assessments. Analytics could predict risk, but interventions were usually advisory rather than operational.

The fourth wave (mid-2010s–early 2020s) used deep learning and multimodal AI. Automated essay scoring, code-feedback systems, speech-based tutoring, and vision-based engagement analytics improved scale and realism. Yet these tools remained function-level additions to human-run universities.

The fifth wave (2020s–present) is defined by generative AI and agentic large language models. These models can generate coherent explanations, counterexamples, multimodal demonstrations, adaptive tasks, simulated laboratories, rubrics, and Socratic dialogues. The boundary shifts from AI supporting instructional artifacts to AI supporting institutional ecologies. FAI-U represents this next step: a human-governed university where AI generates the environment end-to-end and continuously improves it.

2.2 Empirical and Technical Research Strands Informing FAI-U

Intelligent tutoring meta-analyses indicate that ITS can produce large learning gains approaching one-to-one tutoring effects in constrained domains. However, historical studies also show that authoring costs and domain brittleness constrained deployment beyond narrow tasks. Generative AI weakens this bottleneck by generating tasks and explanations on demand but introduces new risks that require governance.

Learning analytics research demonstrates that predictive models can forecast dropout risk, pacing instability, and mastery gaps early enough to intervene. In most universities these insights remain separate from instruction. FAI-U makes analytics

operational by embedding forecasts into the curriculum engine and personal agents, enabling closed-loop intervention.

Digital-twin studies in higher education increasingly model campuses, labs, and learner trajectories as synchronized simulation environments. This supports proactive optimization of resources and learning pathways, a foundational assumption of the AI Campus Core and Learner Twin layers.

Multi-agent LLM research shows that role-specialized agent ecologies improve robustness, controllability, and pedagogical richness. Education-oriented multi-agent frameworks demonstrate feasibility of AI professors, TAs, librarians, and facilitators co-inhabiting learning environments.

Assessment research in the GenAI era emphasizes that written-product evaluations are increasingly vulnerable to undetectable AI assistance. Authentic assessment formats oral Vivas, in-situ design tasks, and portfolios are therefore becoming dominant proposals for integrity-resilient graduate evaluation. This motivates FAI-U's Socratic oral backbone.

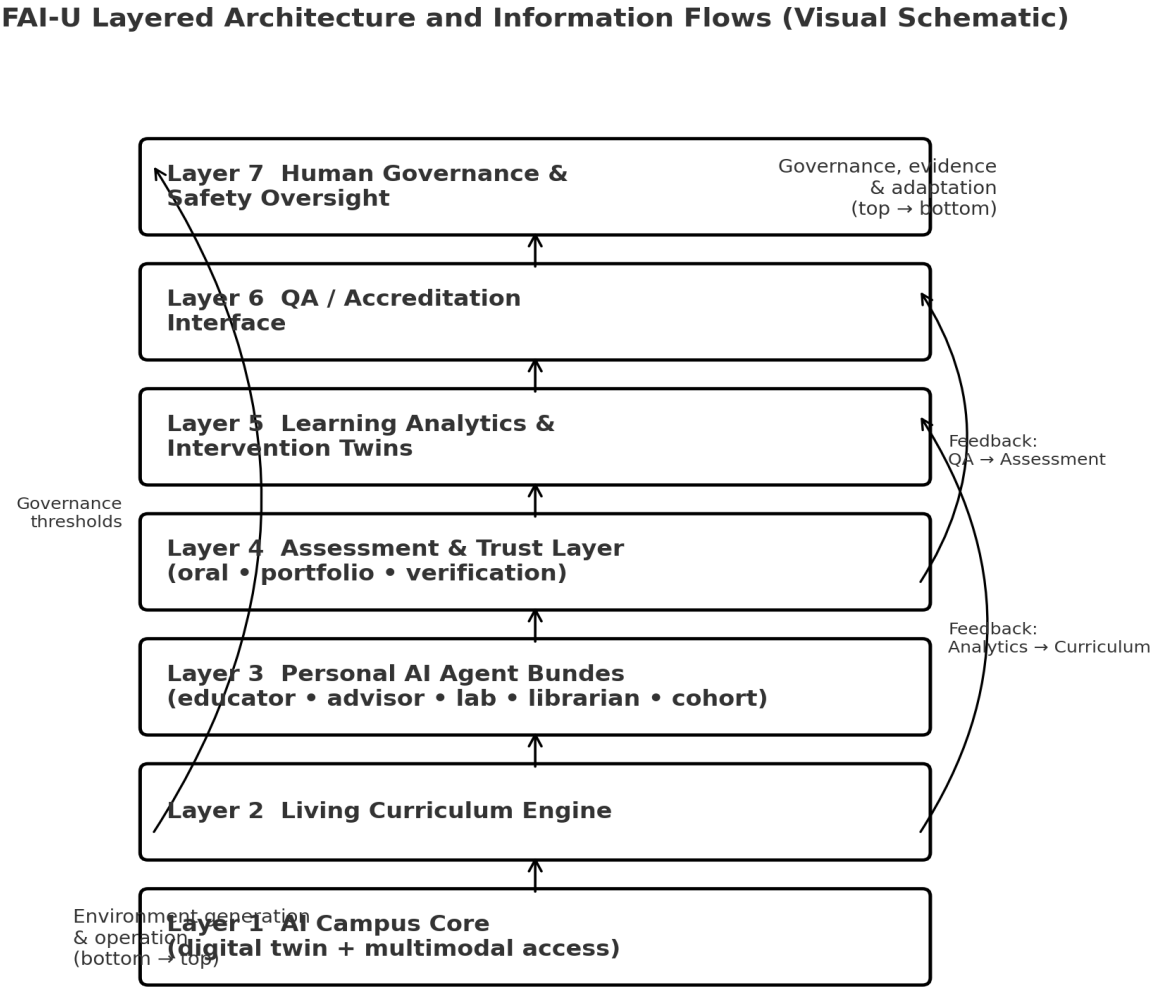
3. Theoretical Foundations for a Fully AI-Generated University

FAI-U is anchored in three theory families. First, outcome- and competency-based graduate education: stable outcomes defined by faculty are the core of academic authority, while AI dynamically generates multiple individualized pathways. Second, self-regulated learning and metacognition: graduate learners must plan, monitor, and adapt, so FAI-U embeds pacing forecasts, reflective prompts, and wellbeing-aware interventions. Third, epistemic cognition and communities-of-practice: graduate learning is identity formation through participation in disciplinary discourse; thus FAI-U includes AI peer cohorts, frontier colloquia, and research studios that reproduce scholarly community membership.

4. The FAI-U Layered Model

FAI-U is a university-scale system in which AI generates and runs the learning environment end-to-end, while human governance defines aims and constraints. Seven interacting layers structure the system: (1) AI Campus Core, (2) AI Instructional Design Engine, (3) Personal AI Agent Bundles, (4) Assessment & Trust Layer, (5) Learning Analytics & Intervention Twins, (6) Quality Assurance & Accreditation Interface, and (7) Human Governance & Safety Oversight. Layering preserves modularity and traceability, enabling autonomy without institutional drift.

Figure 3. *FAI-U layered architecture and information flows.*



4.1 AI Campus Core: Digital Twin University

The AI Campus Core is a procedurally generated digital twin of the university. It includes academic spaces (lecture studios, seminar rooms, exam chambers), research spaces (methods clinics, lab benches, thesis studios), and social spaces (cohort commons, interdisciplinary cafés). When new program needs or research themes emerge, the campus core generates or reconfigures dedicated spaces. Multimodal access includes web/mobile and optional VR/AR for embodied labs, supporting inclusivity and disciplinary realism.

4.2 Instructional Design Engine: Living Curricula

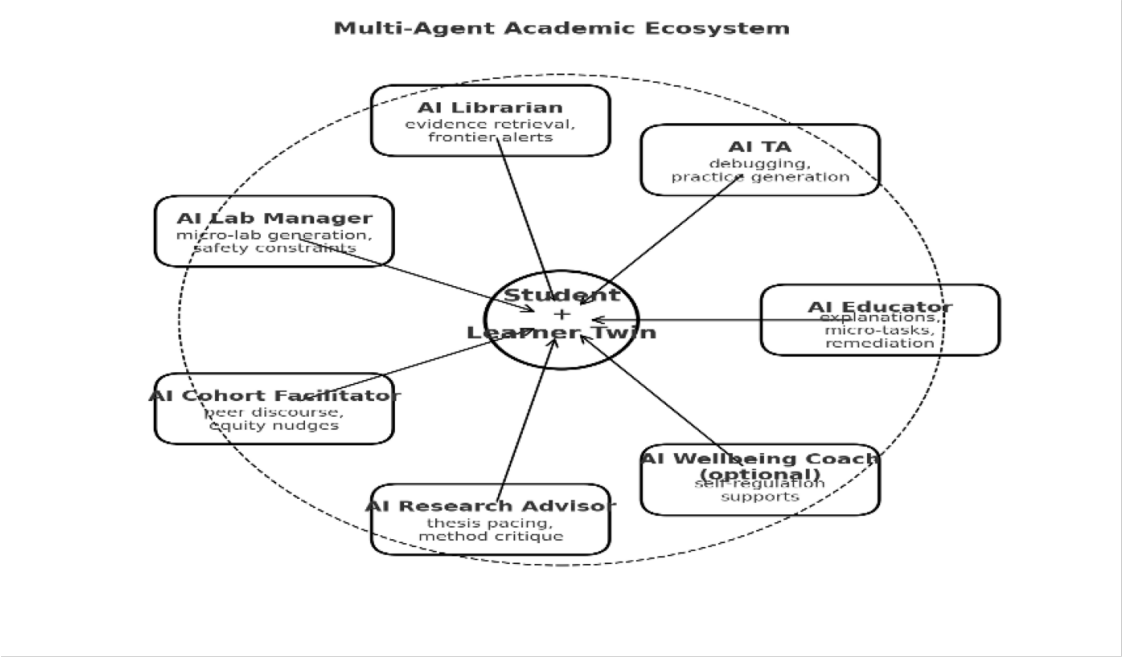
This engine converts human-defined outcomes and accreditation constraints into living curricula. It generates program maps, syllabi, weekly outcomes, readings, micro-labs, apprenticeships, and rubrics. Adaptation is governed by three loops: an evidence loops that grounds content in curated corpora via retrieval-augmented generation; a learner-response loop that uses micro-formative data to adjust sequencing and

representation; and a governance loop that routes high-magnitude changes to human approval.

4.3 Personal AI Agent Bundles

Each graduate learner receives persistent role-specialized agents that coordinate via the learner twin. These include educator agents for conceptual development, TA agents for practice generation and debugging, librarian agents for evidence retrieval and frontier alerts, lab-manager agents for simulation and safety constraints, cohort facilitators for peer discourse, research advisors for thesis milestones, and optional wellbeing coaches. Persistence across semesters allows the system to model a student's conceptual style and research preferences while maintaining a negotiation stance that requires justification rather than supplying final answers.

Figure 4. *Multi-agent academic ecosystem around the learner twin*



4.4 Assessment and Trust Layer

FAI-U integrates continuous micro-assessment, Socratic oral assessment, portfolio evidence, and milestone defenses. Trust mechanisms verify identity and authorship in high-stakes tasks, anchor grading to evidence, monitor demographic and paradigm fairness, and generate explainable rubric traces. Engineering competence is evaluated through modeling, design iteration, failure analysis, and safe deployment reasoning; educational sciences competence through theory choice, validity arguments, methodological identification, and interpretive humility.

Because high-autonomy environments amplify both benefits and risks, FAI-U requires an explicit risk-mitigation architecture. Table 2 summarizes dominant failure modes and maps them to detection signals, controls, and accountable governance units.

Table 2. *Risk–mitigation matrix for FAI-U.*

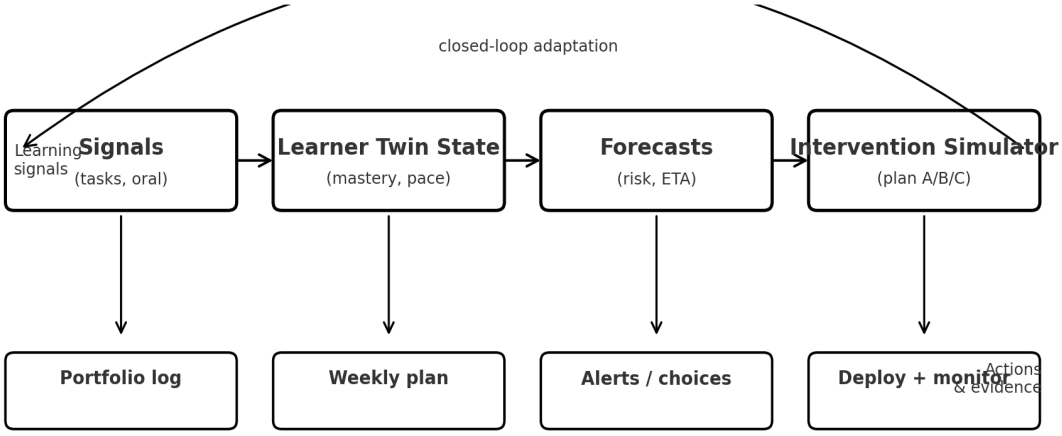
Risk	Description	Detection signals	Mitigation controls	Governance owner
Hallucination	Incorrect generated content	Correction spikes; RAG mismatch	Curated corpora; uncertainty flags; audits	Dept. boards
Demographic bias	Unequal outcomes by group	Parity gaps; complaints	Fairness metrics; human review triggers	University board
Paradigm monoculture	Over-privileging one theory/method	Diversity index drop	Counter-paper requirement; paradigm audits	Dept. boards
Privacy intrusion	Excessive monitoring	Data volume spikes; opt-out demand	Proportional capture; consent gates; retention limits	University board
Over-reliance	Students outsource judgment	Weak justifications; shallow discourse	Socratic vivas; answer-withholding; reflection prompts	Program committees
Credential trust	External distrust of AI grading	Appeals volume; auditor concerns	Rubric traces; evidence logs; co-signed milestones	Accreditation liaison

4.5 Learning Analytics and Intervention Twins

Learning analytics model mastery trajectories, engagement stability, research velocity, burnout risk, collaboration quality, and metacognitive indicators. Learner twins simulate interventions and forecast tradeoffs among mastery, wellbeing, equity, and timely completion. Low-stakes interventions can deploy automatically with consent, while high-stakes flags route to human review. This converts graduate progression from reactive milestone monitoring to predictive pacing.

Figure 5*. Digital-twin learner pacing and intervention loop.*

Figure 4. Digital-Twin Learner Pacing and Intervention Loop (Visual)



4.6 Quality Assurance and Accreditation Interface

FAI-U automates evidence for internal QA and external accreditation. The interface produces aligned syllabi, rubric-change logs, anonymized artifacts, outcome dashboards, bias audits, and improvement reports. Accreditation becomes continuous evidence stream rather than a retrospective compliance task.

4.7 Human Governance and Safety Oversight

Human governance provides legitimacy and safety. Departments define disciplinary outcomes and paradigm diversity rules; university bodies set ethics and privacy policy; external advisory boards represent accreditation and public trust. Governance defines audit schedules for drift and bias, appeals processes, and clear thresholds where AI autonomy must pause for review.

Institutional legitimacy depends on bounded autonomy. Table 3 specifies which curriculum and assessment changes AI may enact autonomously and which require human academic approval.

Table 3. *Governance thresholds for curricular and assessment change.*

Change magnitude	**Examples**	**AI autonomous?**	**Human review required?**
Low	New practice items; alternate explanations	Yes	No
Medium	Insert micro-unit; adjust	Yes, with log	Committee spot-check

	pacing within course		
High	Modify course outcomes; change rubric weights	No	Department board approval
Critical	Alter program outcomes; pass/fail rules	No	University senate + external notice

5. Innovative Graduate-Level Scenarios and Applications

Scenario 1: Living Curriculum in Safe Autonomous Systems. When a frontier verification-by-construction protocol appears mid-semester, the engine retrieves validated sources, generates a micro-lecture, builds counterexample tasks, inserts a simulation lab, and proposes a rubric delta for human approval. Students complete Socratic oral checks probing assumptions, boundary conditions, and failure interpretations.

Scenario 2: AI Micro-Lab for Robotics. A PhD student requests a grasp-planning sandbox robust to occlusion. The lab manager generates tunable occlusion models, sensor noise profiles, industrial object families, adversarial perturbations, reproducibility hashing, and safety limits for sim-to-real transfer. The TA agent enforces baselines and ablations; the professor agent critiques evaluation realism. The system scores experimental decision quality as portfolio evidence.

Scenario 3: Socratic Validity Defense in Educational Sciences. A doctoral student constructing an "engagement" measure defends construct definitions, operationalizations, and validity arguments under AI Socratic questioning. Alternative theoretical accounts are probed, and triangulation is required.

Scenario 4: Digital-Twin Thesis Pacing. Learner twins simulate 12-week plans, forecast milestone readiness, and estimate burnout probability. Rising delay risk triggers multiple feasible trajectories, co-negotiated with the advisor agent.

Scenario 5: Multi-Agent Peer Cohorts. Students are clustered into cohorts by interest, method and skill profile. AI facilitators orchestrate reading circles, debates that enforce epistemic norms, collaborative drafting with reproducibility checklists, and equity-aware discourse nudges.

Scenario 6: AI Methods Clinics on Demand. Two weeks before comprehensive exams, analytics detect shared gaps (e.g., causal inference). The system generates a cohort clinic with micro-lectures, dissertation-context exercises, simulations, and a group viva.

Scenario 7: Research-Ethics Sandbox. For sensitive dissertations, the environment simulates stakeholder perspectives and counterfactual harms, requiring oral defense of safeguards and alignment with IRB principles.

Scenario 8: Industry-Coupled AI Capstone Studio. Real industry briefs are decomposed into competencies and timelines. Teams receive on-demand micro-labs, writing residencies, and oral design reviews; final assessment combines portfolio and viva panels.

The engineering scenarios rely on heterogeneous lab types. Table 4 defines a micro-lab taxonomy that keeps generation aligned to graduate-level competencies such as robustness analysis, adversarial reasoning, sim-to-real transfer, safety policy reasoning, and reproducibility.

Table 4. *Micro-lab taxonomy for graduate engineering.*

Micro-lab type	Purpose	Generated artifacts	Assessment mode
Parameter-sweep lab	Explore sensitivity & robustness	Simulation grids; plots; logs	Portfolio + oral rationale
Adversarial stress lab	Test failure modes under perturbations	Perturbation engines; counterexamples	Socratic defense of realism
Transfer lab	Validate sim-to-hardware transfer	Twin bridge; safety limits	Milestone viva
Safety-policy lab	Check compliance with safety constraints	Formal constraints; traceability logs	Rubric-anchored oral exam
Reproducibility lab	Ensure replicable experiments	Seed control; containerized envs	Automated reproducibility score

6. Discussion: Opportunities, Risks, and Role Transformation

FAI-U enables radical personalization without curricular fragmentation, accelerates research apprenticeship, scales high-quality feedback, and broadens access to authentic

practice. Living curricula remain synchronized to research frontiers; AI micro-labs remove scarcity and permit high-frequency iteration; Socratic oral assessment strengthens reasoning and academic integrity.

Risks include epistemic over-reliance, demographic bias, paradigm monoculture, privacy intrusion, and credential trust concerns. FAI-U mitigates these through justification-first agent policies, continuous fairness and paradigm audits, proportional data governance with consent, and transparent rubric traces with human co-signing of high-stakes milestones.

Faculty roles become more strategic and more human. Faculty act as epistemic supervisors who arbitrate paradigms and approve outcomes, ethical stewards who set privacy and safety constraints, research mentors who cultivate novelty and scholarly identity, and system calibrators who refine corpora and rubrics.

7. Research and Implementation Agenda

Validation must address construct, consequential, ecological, epistemic, and dialogic validity. Construct validity evaluates whether assessments capture intended competencies. Consequential validity studies long-term effects on identity, motivation, creativity, and equity. Ecological validity tests transfer from micro-labs to physical labs and professional practice. Epistemic validity audits curriculum accuracy and paradigm balance. Dialogic validity evaluates whether Socratic assessments probe reasoning rather than surface fluency.

Evaluation requires longitudinal mixed methods: analytics for mastery trajectories and pacing, expert audits of artifacts, discourse analysis of cohort dialogues, and interviews studying scholarly identity. Engineering pilots should include sim-to-real replication. Educational-science pilots should replicate AI-native validity studios across contexts.

Safety-by-design includes uncertainty signaling, human-review thresholds, auditability of content and rubric drift, demographic and paradigm fairness monitoring, privacy proportionality, and agent sandboxing. Accreditation pilots should map AI outputs to outcomes, give auditors access to evidence streams, and preserve human appeals.

To guide empirical pilots and accreditation evidence, Table 5 presents the validity evaluation matrix proposed for FAI-U.

Table 5. *Validity evaluation matrix for FAI-U.*

Validity type	Primary question	Methods	Evidence
Construct	Does assessment measure competence?	Expert rubric audits; modeling	Oral scores vs. artifacts
Consequential	What long-term effects occur?	Longitudinal surveys & interviews	Identity/motivation shifts
Ecological	Does learning transfer?	Field tests; hardware trials	Sim-to-real performance
Epistemic	Is curriculum accurate & diverse?	Source audits; paradigm indices	Error/diversity logs
Dialogic	Is reasoning deeply probed?	Discourse analysis; think-alouds	Justification quality

8. Conclusion

The Fully AI-Generated University Learning Environment (FAI-U) advanced in this chapter represents a shift in higher education that is qualitative rather than incremental. Instead of treating artificial intelligence as a supplement to traditional university operations, FAI-U positions AI as the generative substrate of the learning environment itself—capable of continuously producing curricula, instruction, laboratories, assessment conditions, feedback cycles, and institutional evidence, while remaining constrained by explicit human governance. In this sense, FAI-U is not a "technologized classroom," but a redesigned university ecology: a system that can evolve in real time without losing legitimate academic control.

This redesign directly addresses structural limitations that have long challenged graduate education. Conventional universities, even when digitized, rely on low-frequency feedback, fixed curricular pacing, scarce laboratory access, and supervision models that scale poorly. These constraints are especially costly in graduate engineering, where competence emerges from repeated design–test–revise cycles, and in educational sciences, where theoretical reasoning and validity-based critique require sustained dialogic apprenticeship. By enabling on-demand AI micro-labs, living curricula synchronized to

research frontiers, and persistent multi-agent mentorship ecosystems, FAI-U expands the bandwidth of graduate learning without lowering the epistemic standards that define university credibility.

The chapter also emphasizes that the critical challenge is not the raw capability of generative AI but the institutional architecture required to govern it safely. Universities exist as trust-bearing organizations. Their social mandate depends on the reliability of credentials, fairness of evaluations, integrity of research training, and protection of students' autonomy and privacy. FAI-U therefore embeds governance not as a peripheral policy layer but as a core system logic: high-impact curricular shifts require departmental authorization, assessment changes are auditable through rubric traces, paradigmatic diversity is monitored to prevent epistemic monoculture, and privacy operates through proportional data capture rather than surveillance by default. These safeguards are essential for converting AI-native education into an academically legitimate and publicly defensible model.

A central implication of FAI-U is the transformation of roles rather than their elimination. Faculty remain the epistemic stewards of disciplines, but their labor migrates from routine delivery toward higher-value work: defining threshold concepts, supervising frontier integration, adjudicating methodological legitimacy, mentoring research identity, and monitoring drift in the learning ecosystem. Students, correspondingly, are not reduced to consumers of automated explanations. Instead, they are pushed toward deeper agency by justification-first tutoring and dialogic Socratic assessments that require defensible reasoning, uncertainty calibration, and methodological ownership. In a mature FAI-U environment, the most valued graduate predicate is not fluent answer production but demonstrable epistemic judgment.

At the same time, the chapter does not assume that AI-generated universities will be automatically beneficial. Institutional power amplified by AI can produce new risks if left unchecked. Over-reliance may weaken independent scholarly identity; bias may reproduce inequities at scale; model drift may gradually distort outcomes; and excessive personalization may fragment shared academic standards. The chapter's layered architecture, multi-agent specialization, and validity-driven research agenda are therefore presented as necessary conditions for responsible adoption rather than optional enhancements. A fully AI-generated environment must be as measurable and contestable as it is adaptive and scalable.

Finally, FAI-U suggests a broader historical interpretation: universities have repeatedly evolved through technological and epistemic revolutions—printing, industrial laboratories, mass higher education, the internet—each time renegotiating the boundary between knowledge, institution, and society. Generative AI introduces the first revolution in which the learning environment itself becomes dynamically producible. If governed with rigor, pluralism, and humility, FAI-U can become a platform not only for scaling graduate

teaching but for accelerating research socialization, widening access to frontier practices, and strengthening the consistency of academic quality across diverse contexts. The decisive question is therefore not whether universities will use AI, but **what kind of universities they will become through AI**—and whether they will retain their defining ideals of truth-seeking, fairness, human development, and public trust while entering an era of continuously generated education.

References

Alkhatlan, A., & Kalita, J. (2019). Intelligent tutoring systems: A comprehensive historical survey with recent developments. International Journal of Computer Applications, 181(43), 1–17.

Attaran, M., & Celik, B. G. (2023). Digital twins and their applications in education: A review of frameworks, benefits, and challenges. Educational Technology Research and Development.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational Researcher, 13(6), 4–16.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. Journal of the Learning Sciences, 2(2), 141–178.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18(1), 32–42.

Chen, N.-S. (2025). Harnessing large language models for education: A framework for pedagogical AI agents. In E-Learning and Enhancing Soft Skills (pp. 13–35). Springer.

Chi, M. T. H. (2009). Active-constructive-interactive: A conceptual framework for learning activities. Topics in Cognitive Science, 1(1), 73–105.

Corbett, A. T., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring systems. In Handbook of human-computer interaction (pp. 849–874). Elsevier.

Creswell, J. W., & Creswell, J. D. (2018). Research design: Qualitative, quantitative, and mixed methods approaches (5th ed.). SAGE.

Graesser, A. C., Conley, M., & Olney, A. (2012). Intelligent tutoring systems. In APA educational psychology handbook (pp. 451–473). APA.

Grieves, M. (2003). Digital twin: Manufacturing excellence through virtual factory replication. University of Michigan Lecture Series.

Gulikers, J. T. M., Bastiaens, T. J., & Kirschner, P. A. (2004). A five-dimensional framework for authentic assessment. Educational Technology Research and Development, 52(3), 67–86.

Howard, S. K., & Mozejko, A. (2022). Authentic assessment in higher education: A systematic review. Assessment & Evaluation in Higher Education, 47(5), 653–670.

Jiang, Y.-H., Li, R., Zhou, Y., Qi, C., Hu, H., Wei, Y., Bo, J., & Wu, Y. (2024). AI agent for education: Von Neumann multi-agent system framework. arXiv preprint arXiv:2501.00083.

Koedinger, K. R., & Corbett, A. (2006). Cognitive tutors: Technology bringing learning sciences to the classroom. In The Cambridge handbook of the learning sciences (pp. 61–77).

Kulik, J. A., & Fletcher, J. D. (2016). Effectiveness of intelligent tutoring systems: A meta-analytic review. Review of Educational Research, 86(1), 42–78.

Lave, J., & Wenger, E. (1991). Situated learning: Legitimate peripheral participation. Cambridge University Press.

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). Intelligence unleashed: An argument for AI in education. Pearson.

Nwana, H. S. (1990). Intelligent tutoring systems: An overview. Artificial Intelligence Review, 4, 251–277.

Panadero, E. (2017). A review of self-regulated learning: Six models and four directions for research. Frontiers in Psychology, 8, 422.

Palmer, A., Smith, J., & Lee, R. (2021). Virtual laboratories as digital twins for engineering education. Journal of Engineering Education, 110(4), 789–812.

Pintrich, P. R. (2004). A conceptual framework for assessing motivation and self-regulated learning in college students. Educational Psychology Review, 16(4), 385–407.

Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. EDUCAUSE Review, 46(5), 30–40.

Suppes, P. (1966). The uses of computers in education. Scientific American, 215(3), 206–220.

Williamson, B., & Eynon, R. (2020). Historical threads, missing links, and future directions in AI in education. Learning, Media and Technology, 45(3), 223–235.

Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. Theory Into Practice, 41(2), 64–70.

GEÇİCİ KAPAK

Kapak tasarımı
devam ediyor.