Scientific publishing house
Professional Bulletin

Information Technology and Security

Issue №2/2025

A scientific journal for the best specialists in the industry. Inside: original works on AI, data analysis, cloud technologies and IT innovations.

INDEXATIONS

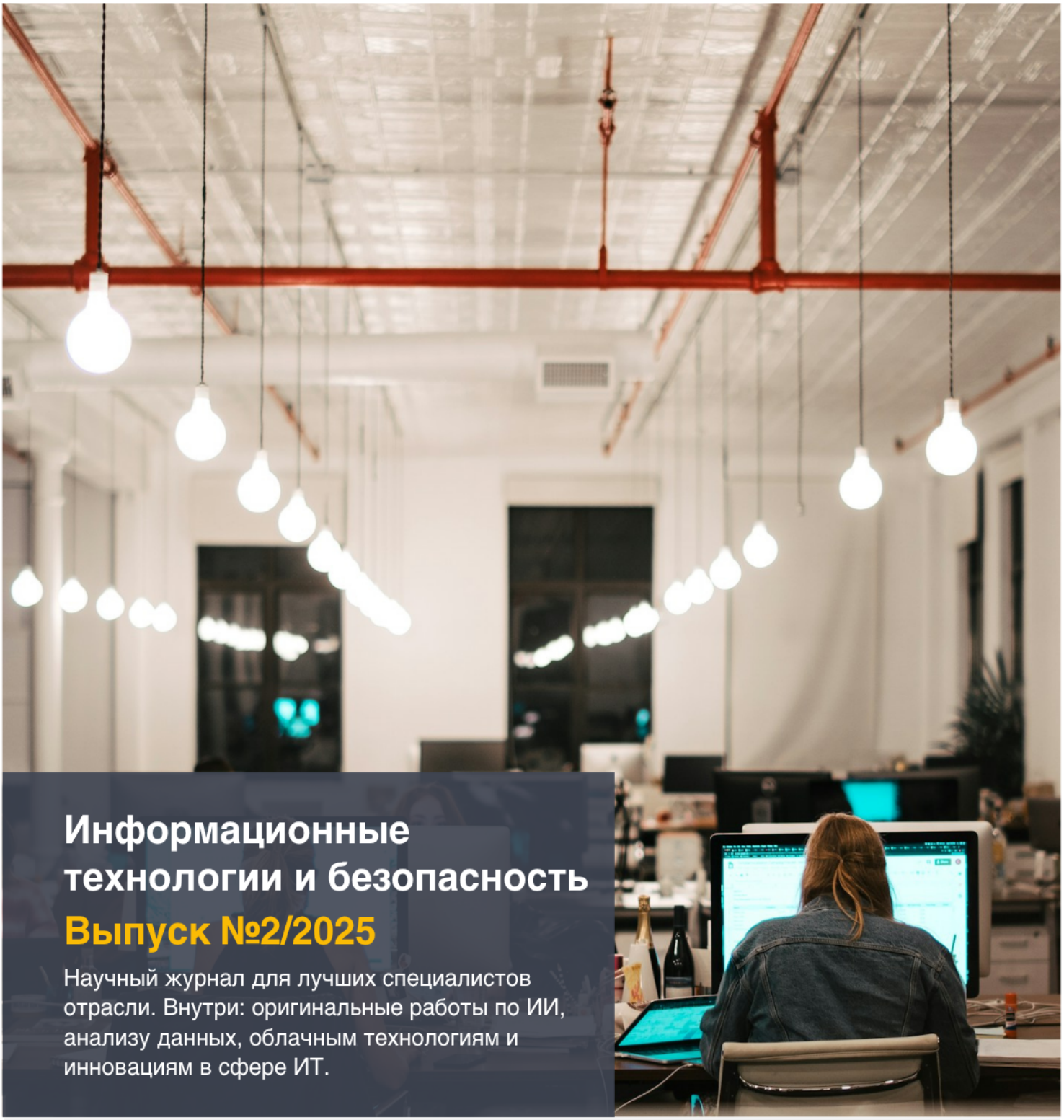
Google Scholar · INTERNATIONAL Scientific Indexing · Academic Resource Index ResearchBib · НАУЧНАЯ ЭЛЕКТРОННАЯ БИБЛИОТЕКА eLIBRARY.RU · CYBERLENINKA · CiteFactor Academic Scientific Journals

ISSN 3100-444X

support@professionalbulletinpublisher.com
professionalbulletinpublisher.com/

Brasov, Sat Sanpetru, Comuna Sanpetru, Str. Sfintii Constantin si Elena, nr. 6

Научное издательство
Профессиональный Вестник



Информационные технологии и безопасность

Выпуск №2/2025

Научный журнал для лучших специалистов отрасли. Внутри: оригинальные работы по ИИ, анализу данных, облачным технологиям и инновациям в сфере ИТ.

ИНДЕКСАЦИИ ЖУРНАЛА

Google Scholar · International Scientific Indexing · Academic Resource Index ResearchBib · Научная электронная библиотека eLIBRARY.RU · CyberLeninka · CiteFactor Academic Scientific Journals

ISSN 3100-444X

support@professionalbulletinpublisher.com

professionalbulletinpublisher.com/

Brasov, Sat Sanpetru, Comuna Sanpetru, Str. Sfintii Constantin si Elena, nr. 6

The scientific publishing house «Professional Bulletin»
Journal «Professional Bulletin. Information Technology and Security»

Professional Bulletin. Information Technology and Security is a professional scientific journal. The publication in it is recommended to practitioners and researchers who seek to find solutions to real-world problems and share their experiences with the professional community. The publication in journal is suitable for those specialists who work and actively develop advanced IT solutions, such as AI, blockchain, big data technologies and others.

The journal reviews all incoming materials. The review is double-blind, carried out by internal and external reviewers of the publishing house. Articles are indexed in a variety of international scientific databases, and access to the journal's database is open to any reader. Publication in the journal takes place 4 times a year.

Publisher's website: https://www.professionalbulletinpublisher.com/

Issue № 2/2025
Brasov County, Romania

Научное издательство «Профессиональный вестник»

Журнал «Профессиональный вестник. Информационные технологии и безопасность»

Профессиональный вестник. Информационные технологии и безопасность – профессиональное научное издание. Публикация в нем рекомендована практикам и исследователям, которые стремятся найти решения для реальных задач и поделиться своим опытом с профессиональным сообществом. Публикация в журнале подходит для тех специалистов, кто работает и активно развивает передовые ИТ-решения, такие как технологии ИИ, блокчейна, больших данных и другие.

Журнал рецензирует все входящие материалы. Рецензирование – двойное слепое, осуществляется внутренними и внешними рецензентами издательства. Статьи индексируются во множестве международных научных баз, доступ к базе данных журнала открыт для любого читателя. Публикация журнала происходит 4 раза в год.

Сайт издательства: https://www.professionalbulletinpublisher.com/

Выпуск № 2/2025

Жудец Брашов, Румыния

Contents

Sapegina V.A.
INTEGRATION OF NEURAL NETWORKS IN CYBERSECURITY OF DISTRIBUTED CLOUD SYSTEMS.....3

Kosheleva E.D., Klychev A.M.
DETECTION OF MALICIOUS ANOMALIES IN IOT-DEVICES USING DEEP LEARNING ...10

Roilian M.
THE EVOLUTION OF WEB ARCHITECTURES: FROM MONOLITHS TO EDGE COMPUTING 19

Khusainov T.Zh.
APPLICATION OF QUANTUM ALGORITHMS IN BIG DATA ANALYSIS.....26

Akhmetshin V.R., Tumashev A.G.
DIGITAL IDENTITY AND DISTRIBUTED LEDGERS IN E-GOVERNMENT SYSTEMS.....33

Andreev G.
DEVELOPMENT OF VISUAL EDITORS FOR DIGITAL MEDIA: ARCHITECTURE, WEBSITE INTEGRATION, AND ADVERTISING POTENTIAL OF INTERACTIVE CONTENT 42

Nasyrova I.N.
MICROSERVICE ARCHITECTURES FOR FINANCIAL PLATFORMS: CHALLENGES AND SOLUTIONS 49

Bastrykin Y.L., Yumasheva N.B.
USE OF GRAPH DATABASES FOR USER BEHAVIOR ANALYSIS 56

Zhukovets L.I.
NEURAL NETWORK ROBUSTNESS TO ADVERSARIAL ATTACKS IN MEDICAL SYSTEMS 66

Geitz N.
EQUIPMENT FAILURE PREDICTION MODELS BASED ON FUSION-ALGORITHMS.....73

Содержание выпуска

Sapegina V.A.
INTEGRATION OF NEURAL NETWORKS IN CYBERSECURITY OF DISTRIBUTED CLOUD SYSTEMS 3

Kosheleva E.D., Klychev A.M.
DETECTION OF MALICIOUS ANOMALIES IN IOT-DEVICES USING DEEP LEARNING ... 10

Roilian M.
THE EVOLUTION OF WEB ARCHITECTURES: FROM MONOLITHS TO EDGE COMPUTING 19

Khusainov T.Zh.
APPLICATION OF QUANTUM ALGORITHMS IN BIG DATA ANALYSIS 26

Ахметшин В.Р., Тумашев А.Г.
ЦИФРОВАЯ ИДЕНТИЧНОСТЬ И РАСПРЕДЕЛЁННЫЕ РЕЕСТРЫ В E-GOVERNMENT СИСТЕМАХ 33

Andreev G.
DEVELOPMENT OF VISUAL EDITORS FOR DIGITAL MEDIA: ARCHITECTURE, WEBSITE INTEGRATION, AND ADVERTISING POTENTIAL OF INTERACTIVE CONTENT 42

Nasyrova I.N.
MICROSERVICE ARCHITECTURES FOR FINANCIAL PLATFORMS: CHALLENGES AND SOLUTIONS 49

Bastrykin Y.L., Yumasheva N.B.
USE OF GRAPH DATABASES FOR USER BEHAVIOR ANALYSIS 56

Жуковец Л.И.
УСТОЙЧИВОСТЬ НЕЙРОСЕТЕЙ К ЦЕЛЕНАПРАВЛЕННЫМ АТАКАМ В МЕДИЦИНСКИХ СИСТЕМАХ 66

Geitz N.
EQUIPMENT FAILURE PREDICTION MODELS BASED ON FUSION-ALGORITHMS 73

UDC 004.8

INTEGRATION OF NEURAL NETWORKS IN CYBERSECURITY OF DISTRIBUTED CLOUD SYSTEMS

Sapegina V.A.
master's degree, University of Twente (Enschede, Netherlands)

ИНТЕГРАЦИЯ НЕЙРОСЕТЕЙ В СИСТЕМЫ КИБЕРБЕЗОПАСНОСТИ РАСПРЕДЕЛЁННЫХ ОБЛАКОВ

Сапегина В.А.
магистр, Университет Твенте (Энсхеде, Нидерланды)

Abstract

This article explores the integration of neural networks into cybersecurity frameworks for distributed cloud systems. The study examines architectural requirements, deployment strategies, model robustness, and resilience to adversarial attacks. Particular attention is paid to real-time anomaly detection, scalability, and the challenges of maintaining model integrity across decentralized environments. The proposed approaches demonstrate the effectiveness of neural models in enhancing threat detection and adaptive defense mechanisms in complex cloud infrastructures.

Keywords: neural networks, cybersecurity, distributed cloud systems, anomaly detection, adversarial training, model robustness, threat intelligence, scalability.

Аннотация

Статья посвящена интеграции нейросетевых моделей в системы обеспечения кибербезопасности распределённых облачных сред. Рассматриваются архитектурные особенности, стратегии развёртывания, устойчивость моделей к атакам и их способность к адаптивному обнаружению аномалий в реальном времени. Отдельное внимание уделено вопросам масштабируемости и защиты модели от внешнего воздействия в условиях децентрализованной инфраструктуры. Представленные подходы демонстрируют эффективность нейросетей в контексте обеспечения интеллектуальной и устойчивой защиты облачных систем.

Ключевые слова: нейросети, кибербезопасность, распределённые облака, обнаружение аномалий, адверсариальное обучение, устойчивость модели, интеллектуальная защита, масштабируемость.

Introduction

In the context of rapidly expanding digital infrastructures, distributed cloud systems have become a fundamental component of contemporary information technologies. These systems ensure scalability, resilience, and operational flexibility by decentralizing data processing across geographically dispersed nodes. However, their open and dynamic architecture introduces new vectors for cyber threats that traditional security mechanisms are often insufficient to address. Increasingly complex attacks such as Advanced Persistent Threats (APTs), zero-day exploits, and multi-stage intrusions exploit the distributed nature of cloud environments, targeting vulnerabilities across services, interfaces, and orchestration layers.

Against this backdrop, neural networks (NNs) have gained attention for their ability to recognize patterns and anomalies in high-dimensional datasets, making them promising candidates for augmenting cybersecurity mechanisms. Unlike rule-based systems, which require predefined heuristics, NNs can learn from historical data and adapt to evolving attack patterns. Their use in

intrusion detection, behavioral profiling, and predictive threat modeling has demonstrated superior performance, particularly in scenarios where traditional systems produce high false-positive rates or fail to generalize across diverse threat landscapes.

The present study aims to examine the integration of neural networks into cybersecurity frameworks specifically tailored for distributed cloud systems. The objective is to systematize architectural approaches, analyze the efficiency of various NN-based detection methods, and identify key challenges associated with deployment, such as latency, model drift, and data privacy concerns. The results are expected to inform the development of adaptive, intelligent, and scalable defense strategies aligned with the architectural characteristics of distributed computing infrastructures.

Main part. Conceptual framework for neural network-based cybersecurity in distributed cloud systems

Ensuring robust cybersecurity in distributed cloud systems necessitates a departure from static, signature-based defense models toward intelligent, adaptive solutions. Neural networks, as a class of machine learning algorithms capable of identifying complex nonlinear dependencies, present an opportunity to redesign core security logic in such systems. Rather than treating security as a peripheral concern, modern architectures are increasingly embedding intelligent modules into the operational layers of infrastructure.

As illustrated in figure 1, neural networks are positioned as a foundational enabler of cybersecurity across distributed systems and cloud environments. This conceptual model reflects the essential role of NNs in enabling automated analysis of system behavior, detection of anomalous activity, and adaptive mitigation strategies. Cybersecurity, in this context, is not treated as a static perimeter but as an evolving, intelligent process informed by continuous data-driven inference.

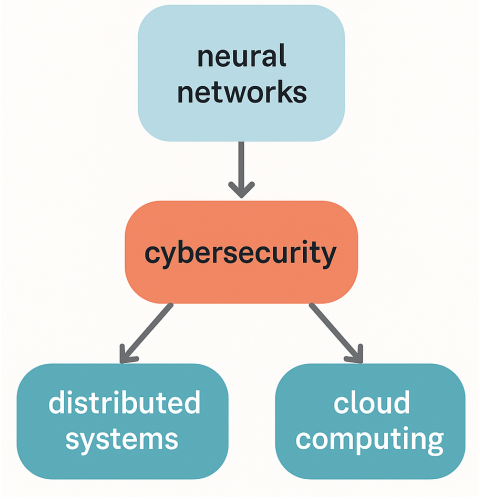


Figure 1. Integration of neural networks into cybersecurity architecture for distributed cloud systems

The interplay between these layers-neural intelligence, security enforcement, and distributed computing-forms a feedback loop where observations lead to predictions, predictions to actions, and actions to new data. By structuring defense in this way, organizations can transition from reactive incident response to proactive risk anticipation. This model also accommodates the volatility and scale of distributed architectures, where static rule sets and pre-configured filters prove insufficient. Neural network models, trained on system logs, traffic metadata, or behavioral patterns, offer dynamic detection capacity that aligns with the flexibility required by modern cloud systems [1].

Neural network implementation for anomaly detection in cloud system logs

Anomaly detection is a core component of modern security frameworks, particularly in the context of distributed cloud environments where telemetry data is abundant and diverse. NNs are increasingly utilized for this task due to their capacity to generalize from high-dimensional inputs and detect patterns indicative of abnormal system behavior. These models are especially effective in scenarios involving structured logs, where statistical and temporal features can be extracted to inform classification.

The code sample below presents a basic implementation of a feedforward NN using the PyTorch library. The model is trained to distinguish between normal and anomalous log events based on

numerical features, enabling integration into security pipelines for real-time or batch-based inference [2].

```
import torch
import torch.nn as nn

class AnomalyDetector(nn.Module):
    def __init__(self, input_size):
        super(AnomalyDetector, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(input_size, 128),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 2)  # Binary classification
        )

    def forward(self, x):
        return self.model(x)
```

The model consists of three fully connected layers with non-linear activation and dropout for regularization. It is compact enough for deployment on edge nodes within the distributed infrastructure, where resource constraints limit the applicability of more complex architectures. When trained on labeled log data, the network can achieve substantial improvements over heuristic or rule-based systems, particularly in detecting novel or obfuscated threats.

Comparative analysis of anomaly detection methods in distributed cloud environments

Security monitoring systems in distributed clouds must contend with high-throughput data streams and fragmented log sources, often requiring automated anomaly detection to operate at scale. Several algorithmic families are typically applied in this context: statistical models, tree-based learners, and NNs [3]. While each class offers specific strengths, their suitability varies depending on the structure, volume, and temporal characteristics of the input data.

Table 1 presents a comparative evaluation of four representative methods commonly employed for anomaly detection in distributed systems. Evaluation metrics include detection accuracy, false-positive rate (FPR), adaptability to evolving threats, and compatibility with distributed deployment models.

Table 1

Comparative characteristics of anomaly detection methods

Method	Accuracy (%)	FPR (%)	Adaptive learning	Scalability
Isolation forest	87.2	11.5	No	Medium
Logistic regression	81.4	8.9	No	High
Feedforward neural net	91.8	6.3	Yes	High
LSTM-based detection	94.1	4.7	Yes	Medium

The table illustrates the superior detection performance of NN-based methods, particularly those using temporal architectures such as LSTM (Long Short-Term Memory). These models demonstrate both higher accuracy and reduced false-positive rates when compared to conventional approaches. Moreover, their ability to learn from streaming data enables continuous refinement without explicit retraining, which is vital in dynamic threat landscapes.

However, trade-offs exist. Temporal models generally incur greater computational overhead and may require batching strategies to handle input sequences effectively. Conversely, simpler models

like Isolation Forests remain useful in constrained environments or as initial filters preceding more sophisticated analysis.

The results underscore the importance of selecting detection algorithms not solely based on accuracy, but also considering deployment constraints and system heterogeneity typical of distributed cloud infrastructures.

Deployment considerations for neural network-based detection in distributed cloud infrastructures

The operational deployment of neural network-based anomaly detection systems in distributed cloud environments poses several practical challenges. Unlike monolithic systems, where models can be trained and applied within a centralized architecture, distributed infrastructures demand careful orchestration of model hosting, data streaming, and inference services. These concerns are further amplified by the need to preserve low latency, fault tolerance, and compatibility with heterogeneous compute environments (e.g., edge nodes, containerized microservices, serverless runtimes).

To support distributed inference, NNs are typically embedded into containerized units using frameworks such as TensorFlow Serving or TorchServe, enabling standardized access through RESTful APIs or gRPC endpoints. These containers are then deployed across multiple cloud regions or edge clusters using orchestration platforms like Kubernetes. Model versioning, A/B testing, and automated rollback mechanisms are implemented through CI/CD pipelines to ensure robust lifecycle management. Furthermore, inference can be offloaded to specialized accelerators (e.g., GPUs, TPUs) in nodes with higher capacity, while lightweight fallbacks are used in constrained environments.

Another critical consideration is the trade-off between local and centralized detection. In some architectures, log streams are forwarded to a central analysis engine, where models are executed on aggregated data to ensure global context and minimize false positives. However, this approach increases network traffic and introduces latency. Alternatively, on-device or edge inference allows for immediate threat recognition and localized response, which is particularly relevant for real-time applications or systems operating in bandwidth-limited settings. Hybrid schemes-where simple models execute at the edge and complex classifiers operate centrally-are increasingly favored to balance latency, accuracy, and system load [4].

Additionally, security and trust in the model itself become important in adversarial environments. Model poisoning, inference manipulation, or reverse engineering of the detection mechanism are feasible threats if the deployment lacks appropriate hardening. Therefore, secure enclaves, encrypted model transmission, and model fingerprinting are recommended in high-risk scenarios. In sensitive applications, federated learning may also be considered as a privacy-preserving strategy, enabling decentralized training across nodes without sharing raw data [5].

Threat vectors targeting neural network models in distributed cloud security systems

As neural networks become integral components of cloud-based cybersecurity frameworks, they themselves become targets of adversarial exploitation. Threat actors increasingly shift their focus toward the machine learning (ML) layer, leveraging architectural and training-time vulnerabilities to undermine the detection capabilities of the models. In distributed cloud systems, where models may be exposed through APIs or run across multiple untrusted nodes, the attack surface expands significantly.

Figure 2 outlines the primary categories of attacks on neural networks that pose critical risks in the context of distributed architectures. These include adversarial examples, in which malicious inputs are designed to mislead the model; data poisoning, where training datasets are manipulated to induce biased or dysfunctional behavior; and broader systemic vulnerabilities introduced through the distributed nature of cloud-based ML.

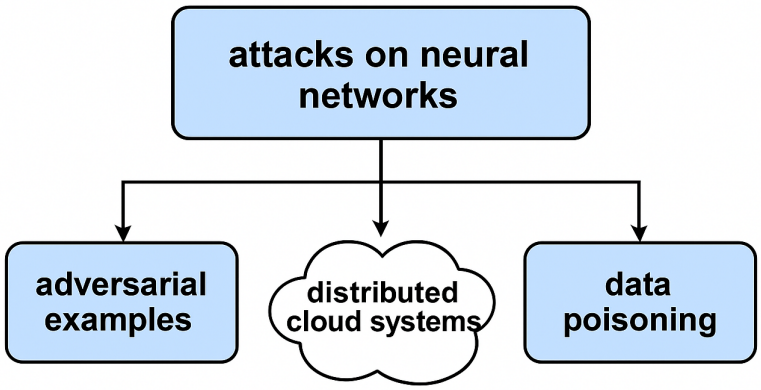


Figure 2. Types of attacks on neural network models in distributed cloud systems

Adversarial examples involve minimally perturbed inputs that appear benign to humans but cause misclassification in neural networks. Such attacks can be mounted in real time against API-accessible services or through tampering with input streams. These perturbations are often imperceptible yet exploit the model's sensitivity to specific dimensions of the input space.

Data poisoning refers to the intentional contamination of training data to compromise a model's learning process. In distributed cloud systems where federated or multi-tenant training is used, poisoned samples may be introduced at edge nodes or external data sources [6]. This method not only degrades detection accuracy but may embed stealthy failure modes within the model.

Distributed attack surfaces further complicate defense, as models may be replicated or fragmented across cloud nodes. This increases the risk of partial model exposure, gradient leakage, and unauthorized model extraction. Without proper encryption, isolation, and access controls, these distributed deployments may inadvertently expose the security layer to exploitation.

As shown in figure, these threat vectors converge around the model level, necessitating a shift in cybersecurity architecture from merely protecting data and infrastructure to actively defending the ML components themselves. This requires the implementation of adversarial training, robust model validation, differential privacy techniques, and continuous monitoring of model outputs under uncertainty.

Adversarial training as a defense strategy for cloud-based neural models

As neural networks are increasingly deployed in security-sensitive applications, their vulnerability to adversarial manipulation poses a significant threat. In distributed cloud systems, where models are exposed through public interfaces or operate on partially observable data, this risk is amplified. One of the most effective defense strategies developed to mitigate such vulnerabilities is adversarial training-a technique where the model is trained not only on clean examples but also on specially crafted adversarial inputs that simulate potential attacks.

The following code demonstrates a simplified adversarial training loop using the Fast Gradient Sign Method (FGSM), a well-known algorithm for generating adversarial examples [7]. This implementation assumes a standard supervised classification setup with labeled data and a PyTorch neural model.

```
import torch
import torch.nn.functional as F

def fgsm_attack(model, data, target, epsilon):
    data.requires_grad = True
    output = model(data)
    loss = F.cross_entropy(output, target)
    model.zero_grad()
    loss.backward()
    data_grad = data.grad.data
    perturbed_data = data + epsilon * data_grad.sign()
    return torch.clamp(perturbed_data, 0, 1)
```

```
def adversarial_training_step(model, data, target, epsilon, optimizer):
    # Generate adversarial examples
    adv_data = fgsm_attack(model, data, target, epsilon)
    # Combine clean and adversarial samples
    combined_data = torch.cat([data, adv_data])
    combined_target = torch.cat([target, target])
    # Forward pass
    output = model(combined_data)
    loss = F.cross_entropy(output, combined_target)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

This method introduces adversarial robustness by systematically exposing the network to perturbed examples during training. The model learns not only from real-world inputs but also from strategically distorted ones, thereby increasing its tolerance to input shifts, noise, and malicious artifacts.

In distributed systems, adversarial training can be implemented either centrally during offline model preparation or dynamically at edge nodes where federated learning is applied [8]. While the latter introduces greater complexity, it ensures localized resilience to region-specific attacks. Additionally, the combination of adversarial training with techniques such as dropout regularization and input normalization further improves generalization under threat conditions.

Conclusion

The integration of neural networks into cybersecurity systems for distributed cloud infrastructures represents a critical evolution in the defense against increasingly complex and adaptive cyber threats. Unlike traditional rule-based mechanisms, NNs provide a scalable and data-driven approach to anomaly detection, intrusion prevention, and behavioral analysis. Their ability to generalize across heterogeneous inputs makes them particularly well-suited for dynamic and decentralized cloud environments.

Throughout this study, several key aspects of such integration have been examined: architectural compatibility, real-time deployment challenges, model performance under operational constraints, and exposure to adversarial threats. The results indicate that, when properly embedded within distributed architectures, NN-based modules not only enhance detection accuracy but also enable autonomous adaptation to evolving threat landscapes.

Nonetheless, the adoption of neural models must be accompanied by robust security measures, including adversarial training, model isolation, and integrity verification. These safeguards are essential to prevent exploitation of the models themselves, especially in publicly accessible or multi-tenant cloud settings. Future work should explore the use of federated learning, secure multi-party computation, and lightweight neural architectures to improve both security and deployability in resource-constrained distributed systems.

References

1. Dey S., Sarma W., Tiwari S. Deep learning applications for real-time cybersecurity threat analysis in distributed cloud systems // World Journal of Advanced Research and Reviews. 2023. Vol. 17. No. 3. P. 1044-1058.
2. Almiani M., Abughazleh A., Jararweh Y., Razaque A. Resilient back propagation neural network security model for containerized cloud computing // Simulation Modelling Practice and Theory. 2022. Vol. 118. P. 102544.
3. Gupta L., Salman T., Ghubaish A., Unal D., Al-Ali A.K., Jain R. Cybersecurity of multi-cloud healthcare systems: A hierarchical deep learning approach // Applied Soft Computing. 2022. Vol. 118. P. 108439.

4. Podder P., Bharati S., Mondal M., Paul P.K., Kose U. Artificial neural network for cybersecurity: A comprehensive review // arXiv preprint arXiv: 2107.01185. 2021.
5. Wang Y., Yang X. Research on enhancing cloud computing network security using artificial intelligence algorithms // arXiv preprint arXiv: 2502.17801. 2025.
6. Awan B. Deep learning neural networks in the cloud // International Journal of Advanced Engineering, Management and Science. 2023. Vol. 9. P. 10.
7. Hasimi L., Zavantis D., Shakshuki E., Yasar A. Cloud computing security and deep learning: An ANN approach // Procedia Computer Science. 2024. Vol. 231. P. 40-47.
Sana M.U., Li Z., Javaid F., Liaqat H.B., Ali M.U. Enhanced security in cloud computing using neural network and encryption // IEEE Access. 2021. Vol. 9. P. 145785-145799.
8. Samriya J.K., Kumar S., Kumar M., Xu M., Wu H., Gill S.S. Blockchain and reinforcement neural network for trusted cloud-enabled IoT network // IEEE Transactions on Consumer Electronics. 2023. Vol. 70. No. 1. P. 2311-2322.

UDC 004.056.5

DETECTION OF MALICIOUS ANOMALIES IN IOT-DEVICES USING DEEP LEARNING

Kosheleva E.D.
master's degree, Kuban state technological university (Krasnodar, Russia)

Klychev A.M.
master's degree, Kuban state technological university (Krasnodar, Russia)

ОБНАРУЖЕНИЕ ВРЕДОНОСНЫХ АНОМАЛИЙ В IOT-УСТРОЙСТВАХ С ПОМОЩЬЮ ГЛУБИННОГО ОБУЧЕНИЯ

Кошелева Э.Д.
магистр, Кубанский государственный технологический университет (Краснодар, Россия)

Клычев А.М.
магистр, Кубанский государственный технологический университет (Краснодар, Россия)

Abstract

Deep learning techniques are increasingly used to detect malicious anomalies in IoT environments, where traditional security mechanisms fail to scale or adapt to dynamic data flows. This study evaluates various neural network architectures suitable for constrained devices, analyzes deployment models from edge to cloud, and examines the resilience of DL systems to adversarial threats. Practical implementation details, including preprocessing pipelines and lightweight inference, are presented alongside comparative performance metrics. Challenges related to data availability, explainability, and system heterogeneity are identified as critical barriers to widespread adoption.

Keywords: IoT security, anomaly detection, deep learning, edge computing, neural networks, adversarial robustness, model deployment, cyber threats.

Аннотация

Глубокие нейросетевые методы находят всё более широкое применение для обнаружения вредоносных аномалий в среде IoT, где традиционные средства защиты оказываются недостаточными. Представлены различные архитектуры нейронных сетей, адаптированные к ресурсным ограничениям устройств, рассмотрены стратегии развёртывания моделей от локального уровня до облака и проведён анализ устойчивости к атакующим воздействиям. Описаны подходы к предобработке данных и организации облегчённого инференса. Особое внимание уделено проблемам доступности данных, объяснимости моделей и разнообразия аппаратных платформ.

Ключевые слова: безопасность IoT, обнаружение аномалий, глубокое обучение, edge-вычисления, нейронные сети, устойчивость к атакам, развёртывание моделей, киберугрозы.

Introduction

The proliferation of Internet of Things (IoT) devices has redefined the architecture of modern digital ecosystems. These devices, deployed across industrial, medical, and consumer domains,

generate continuous streams of sensor data and operational telemetry. Despite their utility, IoT environments are inherently vulnerable to security threats due to limited computational resources, weak authentication schemes, and diverse firmware configurations. Consequently, the detection of malicious anomalies within such systems remains a critical challenge in maintaining the integrity of connected infrastructures.

Traditional rule-based security mechanisms, while effective in constrained scenarios, lack adaptability and scalability when applied to dynamic and heterogeneous IoT networks. Signature-based methods often fail to detect novel or obfuscated attacks, and their performance deteriorates as data volume increases. In contrast, deep learning (DL) techniques offer promising alternatives by enabling the automatic extraction of abstract features and pattern recognition from raw data. When trained on representative datasets, DL models can identify previously unseen malicious behaviors with high accuracy and minimal manual intervention.

The aim of this study is to investigate the applicability of DL-based architectures for detecting malicious anomalies in IoT environments. The research focuses on evaluating neural models for anomaly classification, analyzing deployment constraints specific to edge computing devices, and presenting visual and tabular comparisons of performance metrics. Through this work, we seek to outline the practical considerations and technical advantages associated with using intelligent detection systems in IoT-based infrastructures.

Main part. IoT architecture and threat landscape

The architecture of IoT ecosystems is inherently decentralized, composed of interconnected edge devices, local gateways, and cloud services. Devices often operate with minimal supervision, limited firmware protection, and weak cryptographic modules. These factors, combined with large-scale deployments and wireless communication channels, make IoT infrastructures attractive targets for malicious actors. Attacks may originate from compromised firmware, lateral movement within networks, or spoofed control commands that exploit weak authentication protocols.

Common threat vectors include botnet traffic propagation, firmware injection, distributed denial-of-service (DDoS) attempts, port scanning, and brute-force access to credentials. These anomalies may not immediately disrupt functionality but can significantly compromise data integrity, network availability, and system trustworthiness. Effective detection requires monitoring both network traffic and device-level behavioral signatures [1].

As shown in figure 1, botnet-related anomalies account for the largest portion of malicious activity detected in IoT environments, followed by firmware injection and denial-of-service patterns. This distribution highlights the need for anomaly detection systems that can differentiate subtle variations in traffic patterns and behavioral deviations.

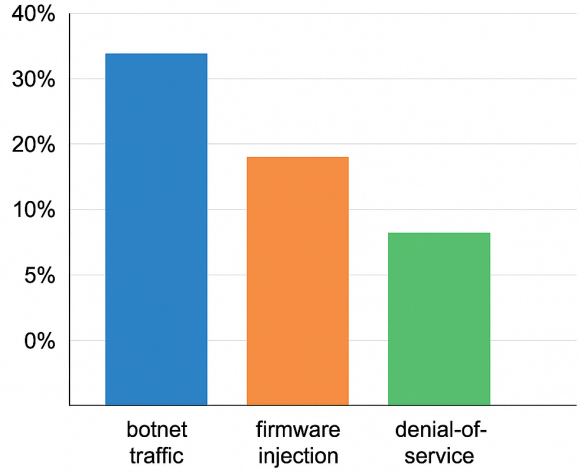


Figure 1. Prevalence of malicious activity types in IoT environments

The figure illustrates the relative frequency of major attack types targeting IoT devices based on recent empirical datasets. Botnet traffic is the most common, reflecting the ease with which devices can be recruited into large-scale coordinated attacks, while firmware injection remains a critical concern due to its persistence and stealth.

Deep learning models for anomaly detection in iot systems

The application of DL methods in the detection of malicious anomalies within IoT systems requires careful alignment between algorithmic complexity and resource constraints. Unlike traditional enterprise networks, IoT devices often operate under limited computational capacity, power restrictions, and connectivity variability. As a result, models deployed in such contexts must balance detection accuracy with efficiency and portability.

Convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and autoencoders are among the most widely adopted DL architectures for anomaly detection in streaming data. CNNs are effective for extracting spatial patterns from preprocessed network traffic features, while LSTMs capture temporal dependencies in sequential telemetry data. Autoencoders, especially in their variational form, can learn compact representations of normal behavior and identify deviations as reconstruction errors [2].

Edge deployment imposes additional constraints: inference latency must remain low, model size must be minimal, and updates must be incrementally applied without complete retraining. To address these issues, model compression techniques such as pruning, quantization, and knowledge distillation are frequently applied after training. Moreover, training itself is typically conducted offline in centralized environments, and only the final inference model is exported to the device.

The example below illustrates a compact feedforward neural network in PyTorch, designed for binary anomaly classification. The model accepts engineered feature vectors derived from packet metadata or system logs and outputs a probability of malicious behavior.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class IoTAnomalyDetector(nn.Module):
    def __init__(self, input_dim):
        super(IoTAnomalyDetector, self).__init__()
        self.fc1 = nn.Linear(input_dim, 64)
        self.fc2 = nn.Linear(64, 32)
        self.dropout = nn.Dropout(0.25)
        self.output = nn.Linear(32, 2)  # 2 classes: normal, anomaly

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = self.dropout(F.relu(self.fc2(x)))
        return self.output(x)
```

This architecture is intentionally shallow and lightweight, making it suitable for execution on edge hardware such as ARM-based microcontrollers or embedded Linux boards. While more complex architectures may yield marginal improvements in accuracy, they do so at the cost of inference speed and energy consumption-two critical parameters in real-time IoT systems [3].

Training such models requires labeled datasets that capture both benign and malicious activity, with an emphasis on generalizability across device types and usage scenarios. Data augmentation and regularization techniques help prevent overfitting, especially when anomaly samples are scarce or imbalanced. Ultimately, the success of DL-based anomaly detection in IoT hinges on its ability to adaptively generalize while remaining lightweight and interpretable.

Model comparison for anomaly detection in IoT environments

The choice of deep learning architecture for anomaly detection in IoT systems must be driven by a balance between classification performance, resource efficiency, and adaptability to streaming conditions. Unlike general-purpose computing environments, IoT deployments operate under highly constrained conditions where even modest increases in model complexity can render real-time detection impractical. As a result, model selection must consider not only accuracy metrics but also inference latency, memory footprint, and robustness to noisy or incomplete data.

Three dominant classes of models are evaluated in this context: feedforward networks (FFNs), recurrent architectures such as LSTM networks, and autoencoder-based anomaly detection systems. FFNs are computationally lightweight and well-suited for tabular input derived from structured logs or metadata. LSTMs, on the other hand, offer improved performance in time-series contexts, where sequence dependencies are essential for capturing temporal anomalies. Autoencoders provide a flexible unsupervised approach, capable of detecting unknown attacks by modeling the normal operational space and identifying deviations as reconstruction errors.

Table 1 presents a comparative analysis of these models based on empirical benchmarks from IoT-specific datasets. The evaluation includes multiple criteria: detection accuracy, false positive rate (FPR), average inference time per sample, and model size (in MB) after compression. All models were trained on the same preprocessed dataset and evaluated using a standardized test protocol.

Table 1

Comparative evaluation of DL models for anomaly detection in IoT

Model	Accuracy (%)	False positive rate (%)	Inference time (ms)	Model size (MB)
Feedforward NN	91.2	5.4	3.2	0.7
LSTM	94.7	3.1	12.8	3.5
Autoencoder	89.6	6.2	5.6	1.1

The results show that LSTM networks achieve the highest overall accuracy (94.7%) and lowest FPR (3.1%), making them ideal for high-integrity anomaly detection in streaming telemetry. However, they exhibit longer inference times and larger model sizes, which may limit their feasibility on low-power devices. FFNs offer a compact and fast alternative, sacrificing a small margin in accuracy for a 4x reduction in latency. Autoencoders provide strong performance in unsupervised scenarios but require careful tuning to avoid underfitting normal patterns or overflagging benign anomalies.

Ultimately, the model choice should align with the deployment target: LSTMs are appropriate for centralized or gateway-based analysis, while FFNs and compressed autoencoders are better suited for edge-level detection on individual IoT devices. Hybrid strategies, in which lightweight models flag suspicious behavior and forward events to a centralized engine for deeper analysis, may offer an optimal balance between responsiveness and detection fidelity.

Deployment strategies for DL-based anomaly detection in IoT

The integration of deep learning models into IoT security infrastructure requires deployment strategies that align with the operational characteristics of the system. Unlike traditional IT environments, IoT networks often exhibit constraints in processing power, memory availability, connectivity bandwidth, and update frequency. Therefore, deployment scenarios must be carefully selected based on the location of inference, required response time, and data sensitivity.

Deployment on a single device (local edge) prioritizes minimal latency and independence from external connectivity. In this configuration, a small feedforward neural network is embedded directly into the device firmware or local runtime environment. This setup is optimal for scenarios requiring millisecond-level decisions, such as real-time access control or actuator response. However, it restricts model complexity and update frequency, making it suitable primarily for known threat profiles.

Gateway-level aggregation offers a trade-off between performance and visibility. Data from multiple devices is aggregated at a local node, where models such as LSTMs can be used to analyze temporal patterns and detect coordinated anomalies [4]. This architecture allows for more powerful detection while maintaining acceptable inference latency and supporting periodic model updates.

Cloud-based central analysis provides the highest analytical power by utilizing sophisticated architectures like autoencoders or transformers on aggregated data. This approach enables continuous model refinement and access to broader datasets, which improves detection quality. However, it

introduces latency and requires reliable connectivity, which may not be suitable for latency-sensitive IoT applications.

Federated learning is an emerging approach that enables distributed training across multiple IoT nodes without sharing raw data. Lightweight models such as pruned FFNs or LSTMs are updated locally and only gradients or model updates are transmitted securely. This model preserves privacy, supports adaptability, and reduces communication overhead-but depends on synchronization and secure update aggregation.

A hybrid cascade detection strategy combines local lightweight models for initial anomaly scoring with selective forwarding of suspicious samples to a centralized analysis engine. This architecture balances response time and accuracy while minimizing data transfer. It is particularly effective in scenarios where bandwidth is limited but detection confidence must remain high.

Table 2 summarizes the comparative characteristics of these deployment strategies across multiple operational dimensions.

Table 2

Deployment strategies for DL-based anomaly detection in IoT

Deployment scenario	**Model type**	**Latency requirement**	**Connectivity dependence**	**Update strategy**
Single-device (local edge)	Feedforward NN	Ultra-low (<5 ms)	Low	Static model
Gateway-level aggregation	LSTM	Low to moderate (5–20 ms)	Medium	Periodic batch update
Cloud-based central analysis	Autoencoder / Transformer	Moderate to high (20–100 ms)	High	Continuous retraining
Federated learning (multi-device)	Compressed FFN or LSTM	Moderate (10–30 ms)	Medium	Secure federated update
Hybrid cascade detection	Lightweight local + full remote	Split latency (2–50 ms)	Variable	Trigger-based escalation

The table highlights the trade-offs associated with each strategy and illustrates how deployment decisions impact model selection, update mechanisms, and latency constraints. No single approach fits all scenarios; rather, deployments must be tailored to the specific constraints and objectives of the target IoT application domain.

Data preprocessing and lightweight inference on IoT edge devices

The effectiveness of DL-based anomaly detection in IoT settings depends not only on the architecture of the model but also on the preprocessing pipeline and the runtime execution strategy. In edge scenarios, both stages must be computationally efficient, modular, and capable of handling incomplete or noisy sensor input.

Preprocessing typically involves normalization, feature extraction, and dimensionality reduction. For structured IoT logs or packet-level telemetry, relevant features include source and destination IP entropy, payload size variability, time between packets, and protocol distribution. These features are computed using sliding windows and prepared in fixed-size vectors suitable for input into a compact neural network [5].

The example below shows a lightweight preprocessing and inference pipeline using PyTorch and NumPy, tailored for execution on embedded Python runtimes such as MicroPython or edge containers.

```
import numpy as np
import torch
import torch.nn.functional as F
from trained_model import IoTAnomalyDetector  # pretrained model class
```

```
# Example input: feature vector from IoT device log
def extract_features(packet_stream):
    avg_payload = np.mean([p['payload_size'] for p in packet_stream])
    std_interval = np.std([p['timestamp_diff'] for p in packet_stream])
    proto_count = len(set(p['protocol'] for p in packet_stream))
    return np.array([avg_payload, std_interval, proto_count], dtype=np.float32)

# Normalize and convert to tensor
def normalize_and_infer(feature_vector, model, mean, std):
    normalized = (feature_vector - mean) / std
    x_tensor = torch.tensor(normalized).unsqueeze(0)
    with torch.no_grad():
        logits = model(x_tensor)
        probabilities = F.softmax(logits, dim=1)
    return probabilities[0][1].item()  # probability of anomaly

# Sample execution
features = extract_features(packet_stream)
model = IoTAnomalyDetector(input_dim=3)
model.load_state_dict(torch.load("iot_detector.pt", map_location='cpu'))
model.eval()

anomaly_score = normalize_and_infer(features, model, mean=np.array([200, 0.5, 4]), std=np.array([100, 0.2, 2]))
print(f"Anomaly probability: {anomaly_score:.3f}")
```

This compact pipeline can be deployed in environments where full-stack frameworks like TensorFlow are too resource-intensive. Once trained and quantized, the model can be stored in less than 1 MB, enabling fast and reliable predictions directly on the device or local gateway. For deployments with intermittent connectivity, the output (anomaly score) can be logged locally or sent asynchronously to a central server only when it crosses a configurable threshold.

By combining efficient feature engineering with optimized inference routines, this setup enables real-time anomaly detection that is both robust and operationally viable across a range of embedded IoT platforms.

Resilience of DL models to adversarial threats in IoT systems

Despite their effectiveness in detecting complex anomalies, DL models deployed in IoT environments are not immune to adversarial threats. In fact, their reliance on learned representations makes them susceptible to subtle manipulations in the input space or model internals, especially when deployed in open or distributed settings. Understanding and mitigating these vulnerabilities is critical to preserving the reliability of anomaly detection systems.

One common attack vector is adversarial perturbation, in which an attacker slightly modifies legitimate inputs to induce misclassification without altering the data in a semantically noticeable way. Feedforward and recurrent models are particularly vulnerable to such perturbations, especially when trained without noise-aware regularization. These attacks can be countered using adversarial training or input smoothing techniques that improve the model's robustness.

Data poisoning attacks occur during the training phase and are especially relevant in federated or decentralized systems. By injecting carefully crafted samples into the training dataset, an attacker can degrade model performance or induce false negatives on malicious patterns. Defensive strategies include robust aggregation mechanisms, outlier detection, and selective validation of incoming data.

More subtle threats include model inversion, which exploits access to outputs or gradients to reconstruct sensitive input data. This attack is most feasible in centralized deployments of lightweight models. Encryption of model parameters and restricted inference access are the primary countermeasures [6].

In network-level scenarios, evasion via protocol mimicry is a growing concern. Attackers craft traffic patterns that imitate benign behavior to avoid detection by models trained on protocol statistics or packet structure. Autoencoders are particularly at risk here, as they often fail to distinguish structurally valid but semantically harmful input. Solutions involve integrating protocol-specific fingerprinting and traffic segmentation into the feature engineering pipeline.

Finally, gradient leakage poses a privacy risk in federated settings. If updates from client devices are not aggregated securely, an attacker can infer training data characteristics from shared gradients. This risk is mitigated through secure aggregation protocols and differential privacy techniques that introduce randomness into updates without significantly compromising learning quality [7].

Table 3 summarizes key attack types, their impact on DL-based IoT detection systems, and corresponding mitigation strategies.

Table 3

Resilience of DL models to adversarial threats in IoT systems

Attack type	Vulnerable models	Impact	Mitigation strategies
Adversarial perturbation	FFN, LSTM	Misclassification of malicious input	Adversarial training, input smoothing
Data poisoning	All (training-stage issue)	Model degradation and false negatives	Robust training, data validation
Model inversion	FFN (centralized deployment)	Exposure of input data patterns	Access control, model encryption
Evasion via protocol mimicry	Autoencoder	Undetected malicious traffic patterns	Traffic fingerprinting, statistical modeling
Gradient leakage	Federated LSTM	Loss of privacy in local updates	Secure aggregation, differential privacy

As the threat landscape evolves, designing resilient learning architectures must become a core consideration in IoT security. This includes not only protecting against direct attacks but also enabling post-deployment model auditing and anomaly explanation to improve transparency and trust.

Practical limitations of DL-based anomaly detection in IoT

While deep learning has demonstrated promising results in detecting malicious activity within IoT infrastructures, its real-world adoption faces several practical constraints. These limitations span technological, organizational, legal, and operational domains, each of which must be addressed to facilitate secure and sustainable deployment [8].

One of the primary challenges is the scarcity of high-quality labeled datasets. Effective supervised training requires large volumes of diverse anomaly-labeled data, which is rarely available in IoT deployments. Many organizations lack centralized logging infrastructure or data retention policies, leading to fragmented, incomplete, or non-standardized datasets. Furthermore, privacy regulations may restrict the collection of raw telemetry data, especially in medical or consumer-facing applications.

Another obstacle is the fragmentation of IoT device ecosystems. Vendors utilize different hardware platforms, operating systems, and communication protocols, resulting in highly heterogeneous environments [9]. DL models trained on one device type or network context may not generalize well to others without additional tuning or retraining. This hampers model portability and increases the cost of cross-platform support.

Regulatory and compliance considerations also play a critical role. For example, anomaly detection models deployed in regulated industries (e.g., healthcare, critical infrastructure) must be explainable and certifiable [10]. Many DL systems operate as “black boxes,” making it difficult to justify security decisions to auditors or regulatory authorities. Efforts to integrate explainable AI into IoT security remain limited and experimental.

In terms of operational maintenance, updating DL models in the field is a non-trivial process. Over-the-air updates may pose security risks themselves, while manual update cycles are labor-intensive and error-prone. Moreover, most IoT devices lack the capability for full retraining or real-time adaptation, requiring the use of transfer learning, distillation, or cloud-assisted update architectures.

Lastly, cost and energy consumption remain persistent concerns. Even with model compression and inference optimization, the deployment of DL components increases hardware complexity, power requirements, and overall system cost-particularly in battery-operated or large-scale sensor deployments.

Addressing these limitations will require advances not only in model architecture but also in system design, regulatory frameworks, data infrastructure, and vendor collaboration [11]. Until these gaps are resolved, the adoption of DL-based anomaly detection in IoT environments will likely be confined to controlled or high-value scenarios.

Conclusion

The increasing ubiquity of IoT devices across critical sectors has intensified the need for effective and scalable anomaly detection systems capable of identifying malicious behavior in real time. Traditional security mechanisms are ill-suited to the dynamic and resource-constrained nature of IoT environments, prompting a shift toward deep learning-based approaches that can learn complex behavioral patterns and generalize to previously unseen threats.

This study explored the architectural and operational dimensions of applying deep learning to anomaly detection in IoT networks. It examined key model types, deployment strategies, and their respective trade-offs in terms of accuracy, latency, and computational overhead. Furthermore, the work highlighted practical challenges including adversarial vulnerability, privacy risks, and the limited availability of labeled datasets for supervised learning. Through comparative analysis and code-level examples, the paper demonstrated how lightweight neural architectures can be effectively adapted to the IoT context while maintaining sufficient robustness and interpretability.

Although deep learning offers significant potential for enhancing IoT security, its successful deployment requires a multidimensional approach that considers infrastructure heterogeneity, regulatory compliance, update logistics, and adversarial resilience. Future research should focus on building adaptive, explainable, and energy-efficient models capable of long-term operation in real-world edge environments. In doing so, DL-based anomaly detection can become a foundational component of next-generation IoT cybersecurity frameworks.

References

1. Abusitta A., de Carvalho G.H., Wahab O.A., Halabi T., Fung B.C., Al Mamoori S. Deep learning-enabled anomaly detection for IoT systems // Internet of Things. 2023. Vol. 21. P. 100656.
2. Riaz S., Latif S., Usman S.M., Ullah S.S., Algarni A.D., Yasin A., Hussain S. Malware detection in internet of things (IoT) devices using deep learning // Sensors. 2022. Vol. 22. No. 23. P. 9305.
3. Diro A., Chilamkurti N., Nguyen V.D., Heyne W. A comprehensive study of anomaly detection schemes in IoT networks using machine learning algorithms // Sensors. 2021. Vol. 21. No. 24. P. 8320.
4. Ullah I., Mahmoud Q.H. Design and development of a deep learning-based model for anomaly detection in IoT networks // IEEe Access. 2021. Vol. 9. P. 103906-103926.
5. Aversano L., Bernardi M.L., Cimitile M., Pecori R., Veltri L. Effective anomaly detection using deep learning in IoT systems // Wireless Communications and Mobile Computing. 2021. Vol. 2021. No. 1. P. 9054336.
6. Ahmad Z., Shahid Khan A., Nisar K., Haider I., Hassan R., Haque M.R., Rodrigues J.J. Anomaly detection using deep neural network for IoT architecture // Applied Sciences. 2021. Vol. 11. No. 15. P. 7050.

7. Dudak A., Israfilov A. Application of blockchain in IT infrastructure management: new opportunities for security assurance // German International Journal of Modern Science. 2024. № 92. P. 103-107.
8. Vigoya L., Fernandez D., Carneiro V., Novoa F.J. IoT dataset validation using machine learning techniques for traffic anomaly detection // Electronics. 2021. Vol. 10. No. 22. P. 2857.
9. Mukherjee I., Sahu N.K., Sahana S.K. Simulation and modeling for anomaly detection in IoT network using machine learning // International journal of wireless information networks. 2023. Vol. 30. No. 2. P. 173-189.
10. Jaramillo-Alcazar A., Govea J., Villegas-Ch W. Anomaly detection in a smart industrial machinery plant using iot and machine learning // Sensors. 2023. Vol. 23. No. 19. P. 8286.
11. Ali S., Abusabha O., Ali F., Imran M., Abuhmed T. Effective multitask deep learning for iot malware detection and identification using behavioral traffic analysis // IEEE Transactions on Network and Service Management. 2022. Vol. 20. No. 2. P. 1199-1209.

UDC 004.75:004.4

THE EVOLUTION OF WEB ARCHITECTURES: FROM MONOLITHS TO EDGE COMPUTING

Roilian M.
Engineering Manager, LeanDNA (Austin, USA)

ЭВОЛЮЦИЯ ВЕБ-АРХИТЕКТУР: ОТ МОНОЛИТОВ К EDGE COMPUTING

Ройлян М.
Engineering Manager, LeanDNA (Остин, США)

Abstract

The article explores the stages in the evolution of web architectures within the context of industrial IT systems – from monolithic and centralized solutions to distributed models, including edge computing. It analyzes architectural and network transformations associated with the shift toward edge-based computation, as well as the impact of these changes on fault tolerance, processing latency, and node autonomy. It is emphasized that the implementation of edge infrastructures requires new approaches to containerization, communication protocols, and security. Special attention is given to the use of edge computing for inventory management, production processes, and integration with ERP/SCADA systems. Practical examples from high-precision manufacturing and logistics sectors are presented. The article underscores the need to rethink architectural principles in the era of industrial digitalization.

Keywords: web architectures, edge computing, containerization, industry, protocols, microservices.

Аннотация

В статье рассматриваются этапы эволюции веб-архитектур в контексте промышленных ИТ-систем – от монолитных и централизованных решений к распределенным моделям, включая edge computing. Анализируются архитектурные и сетевые трансформации, связанные с переходом к вычислениям на периферии, а также влияние этих изменений на отказоустойчивость, задержки обработки и автономность узлов. Подчеркивается, что внедрение edge-инфраструктур требует новых подходов к контейнеризации, протоколам взаимодействия и безопасности. Особое внимание уделено применению edge computing для управления запасами, производственными процессами и интеграции с ERP/SCADA-системами. Представлены практические примеры из отраслей высокоточных производств и логистики. Статья актуализирует необходимость переосмысления архитектурных принципов в условиях цифровизации промышленности.

Ключевые слова: web-архитектуры, edge computing, контейнеризация, промышленность, протоколы, микросервисы.

Introduction

Modern manufacturing companies, especially in high-tech sectors, require fault-tolerant information systems capable of rapid response amid supply chain uncertainty and increasing data volumes from production and warehouse environments. This demand has driven a shift in web architectures – from centralized monoliths to distributed and hybrid models that process data closer to its source. Transitioning from cloud-based systems to edge computing addresses limitations in network bandwidth, communication latency, and the need for uninterrupted operation during

connectivity loss. Edge computing enables local processing, decentralizes decision-making, and enhances the autonomy of digital production platforms, making it especially effective for inventory management, equipment monitoring, and real-time traceability.

The current article aims to explore the evolution of web architectures in the context of their application to industrial IT systems for controlling material flow and inventories, with a focus on the technological and network-specific nature of edge computing. It explores infrastructural details of the solutions like network topology design, interaction among edge nodes and the cloud service, and the potentiality of integrating these along supply chains.

Main part. Technological evolution of web systems

The creation of web applications for the management of industrial and logistics processes represents fundamental shifts in the methods of data processing and transmission, as well as the architecture of software solutions. In the early days of IT infrastructure development, **centralized architectures** based on mainframes and monolithic applications were common. With the advent of **distributed computing**, this was superseded by the use of Service-Oriented Architecture (SOA), followed by **microservice-based solutions**. SOA advocated modularity and reusability of business logic with standardized interfaces, most commonly achieved through SOAP and XML. The evolution towards microservices was a natural continuation of the decomposition trend: applications were refactored into separate components, each doing a specific task and communicating with others using lightweight REST API or message queues.

The next step in the evolution has been **cloud computing**, adopted by many organizations to address scalability, optimize infrastructure costs, and manage centralized services. However, in industrial settings, the above solutions were limited by several critical factors, including latency and external network dependency. The need to balance centralized control with local responsiveness initially led to the development of **hybrid architectures**, which combined cloud-based orchestration with localized processing capabilities. As system demands for real-time decision-making, bandwidth efficiency, and operational autonomy continued to grow, **edge computing** emerged as a logical next step. It advanced the hybrid model by shifting more computational logic directly to edge nodes – closer to sensors, machines, and production lines – thereby enhancing responsiveness, reducing reliance on central connectivity, and enabling more resilient, decentralized industrial systems (table 1).

Table 1

Evolution of web architectures [1, 2]

Architectural model / period of adoption	Key features	Advantages	Disadvantages
Centralized (mainframe / monolith), 1970 – 1990s	Single point of processing; high component coupling; low flexibility.	Centralized control and security.	Low scalability; single point of failure.
Distributed (SOA / microservices) 2000 – 2015	Modularity; reusability of components; standardized interfaces.	Flexibility; component reusability.	Complexity of service coordination; high coupling in ESB
Cloud / hybrid 2010 – present	Scalability; centralized management; resource usage based on demand.	Rapid scaling; cost optimization.	Network latency; dependence on network and cloud providers.
Edge computing 2015 – present	Data processing near the source. Autonomy; low latency.	Minimization of latency. Resilience to connection loss.	Local infrastructure requirements. Complex security assurance infrastructure.

As shown in the table, the evolution of web architectures in the industrial environment represents a gradual transition from centralized, poorly scalable solutions to flexible, fault-tolerant, and adaptive models capable of efficiently processing data near their generation sources. Each architectural paradigm reflected the technological maturity of its time and addressed the current challenges within the existing constraints.

Edge computing as a response to low-latency requirements

Modern manufacturing and logistics processes are increasingly demanding strict requirements for the response time of information systems. This is due to the need for immediate response to changes in equipment status, real-time processing of telemetry from sensors, as well as ensuring synchronization of actions at various levels of the production chain [3]. It is against this landscape that the edge computing model is being developed – an architectural pattern in which computations and decision-making occur close to the source of data, in contrast to remote cloud data centers.

According to Precedence Research, the global edge computing market is estimated to be worth $432,94 billion in 2024 and is expected to reach approximately $5132,29 billion by 2034, growing at an average rate of 28% from 2025 to 2034. North America has remained the market leader in recent years, accounting for about 40%.

The edge computing model is structured to address the growing need for low-latency data processing and real-time decision-making. Three major levels constitute the edge model architecture (fig. 1).

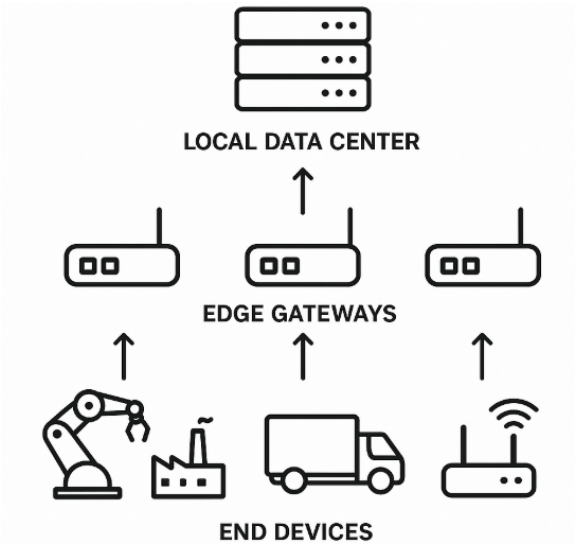


Figure 1. Architecture of edge models

Devices consist of sensors, actuators, and embedded computers that produce raw data streams. **Gateways** act as an intermediate layer, carrying out aggregation, preprocessing, and preliminary data analysis. They can utilize local AI models for making independent decisions and send only aggregated or vital data to the cloud. **Local data centers**, which are on-site or proximate, do more sophisticated tasks, synchronize between nodes, and offer temporary data storage whenever global networks are not accessible [4]. This design minimizes the network load, lowers the response time, and increases the overall system reliability.

The implementation of edge computing as infrastructure demands tight control over network connection properties, particularly in industrial settings where low latency and high availability are paramount. These solutions rely on developing an infrastructure distributed network with **Quality of Service (QoS)** support – a feature through which differentiation of traffic and bandwidth guaranteeing priority data flows are provided (fig. 2).

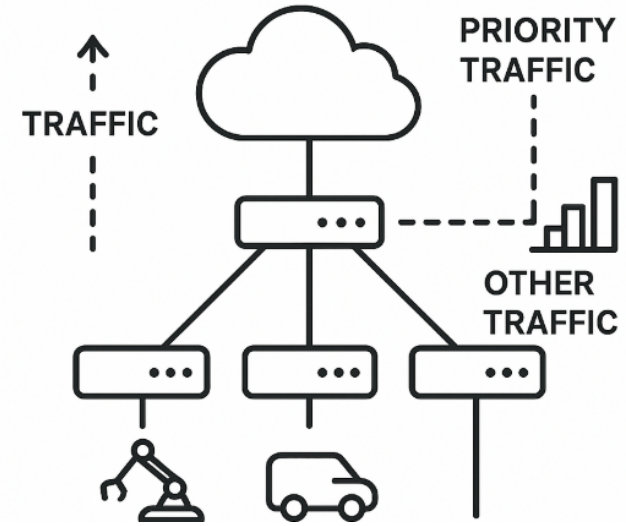


Figure 2. QoS scheme

To ensure scalability and centralized management, the **Software-Defined WAN (SD-WAN)** technology is used. It abstracts the network's logical model from physical transmission channels, implementing routing and policy management at the software level. The global SD-WAN market demonstrates steady annual growth, and according to QKS Group, it is expected to reach $7,04 billion by 2030, with an annual growth rate of 19,43%.

In edge-oriented systems, SD-WAN enables SLA-aware routing, where traffic with low latency tolerance (such as video analytics, data from PLC, and RFID systems) is directed through channels with the lowest RTT (Round-Trip Time) and minimal packet loss. Meanwhile, less critical traffic (such as telemetry for archiving) can be rerouted through channels with higher latency but greater bandwidth (fig. 3).

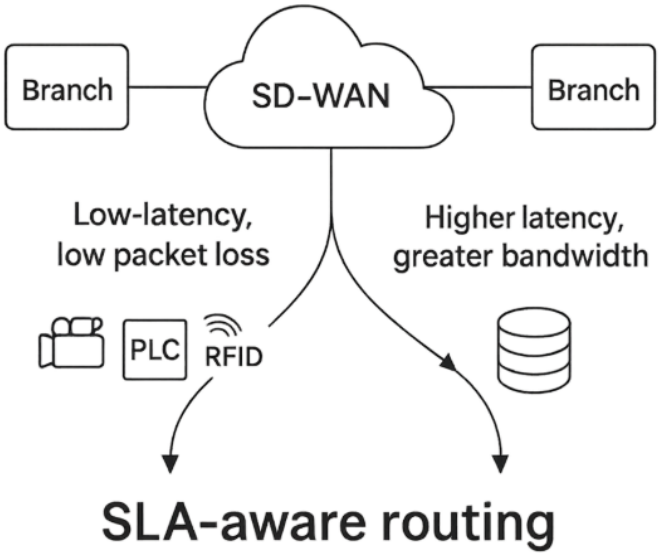


Figure 3. SD-WAN architecture with SLA-aware routing

In addition, the edge infrastructure includes mechanisms such as link redundancy, local breakout, and policy-based forwarding, which are critical in industrial scenarios with high downtime costs. This network architecture not only ensures low data transmission latency but also provides resilience against network degradation or individual link failures, maintaining production continuity and minimizing the risks of disrupting technological cycles.

Technology stack and interaction protocols in edge systems

Containerization and the choice of operating environment are critical in designing edge solutions, as they affect platform stability, scalability, and maintainability. Edge systems typically run on resource-constrained devices, requiring isolated applications with centralized control and secure updates. Lightweight operating systems like Yocto Linux, Ubuntu Core, and BalenaOS, combined with container tools such as Docker and containerd, support efficient, portable

deployments. Orchestration solutions like K3s, MicroK8s, and Portainer enable low-overhead, autonomous edge operations. While containerization ensures modularity and application isolation, it does not fully address lifecycle management, configuration, or system integration. Therefore, an additional runtime layer is needed to handle I/O interaction, event processing, synchronization with central systems, and secure device management.

For example, **Azure IoT Edge Runtime** is a modular runtime environment that enables running Docker containers directly on edge devices. It is integrated with Azure IoT Hub and supports the deployment of both custom logic and built-in modules, including a message router (Edge Hub), modules for streaming analytics, and machine learning models based on ONNX or TensorFlow. Edge Runtime ensures automatic module updates, configuration management, and bidirectional synchronization with the cloud.

Another example is **AWS Greengrass Core**, which offers local execution of AWS Lambda functions, stream processing support, data caching, and offline operation with subsequent synchronization. Within the Greengrass Group, edge devices can exchange messages via a local bus, use MQTT, and perform data transformation before sending it to AWS IoT Core. Special attention is given to security, utilizing X.509 authentication, function isolation, and IAM policies.

The messaging between components of the edge architecture plays a crucial role in **ensuring data consistency**, service manageability, and system resilience in conditions of unstable networks [5]. In the edge environment, priority is given to lightweight and event-driven protocols that can operate with intermittent connectivity and support QoS mechanisms (table 2).

Table 2

Comparison of message exchange protocols in edge architectures [6, 7]

Protocol	Core model	QoS / reliability	Typical applications
MQTT	Publish/subscribe	Three-level QoS (0, 1, 2); resilient to failures.	IoT devices, sensors, edge gateways.
OPC-UA	Client/server + pub/sub	Session support, addressability, built-in PKI.	SCADA, industrial controllers, MES.
REST	Request/response	No QoS, sensitive to latency and disruptions.	API calls, cloud synchronization.
AMQP	Message queue	Delivery confirmation, queues, reliable connection.	Telemetry, corporate data buses.

The choice of data transfer protocol in the edge architecture is determined not only by the characteristics of the network infrastructure but also by the requirements for delivery reliability, the volume of transmitted information, and the type of interacting components. In addition to ensuring reliable data delivery, edge platforms must have the ability to perform local analytics and autonomously respond to events – this requires the inclusion of data processing modules and support for ML inference at the edge.

Event processing on the edge node includes data preprocessing (aggregation, filtering, metric calculation), implementation of business rules, and, if necessary, local decision-making without cloud interaction. Containerized microservices, streaming analytics modules (such as Azure Stream Analytics, Apache Edgent), and ML inference using optimized engines – ONNX Runtime, TensorFlow Lite, OpenVINO – are used to execute models on CPU, GPU, and VPU. In scenarios with high event density, CEP frameworks such as Esper or Apache Flink (in an edge configuration) are employed to detect correlations and respond to complex sequences. When network connectivity is restored, data is synchronized with the central platform through REST, gRPC, or message brokers, considering QoS and retransmission logic, ensuring data consistency and integrity across the system.

In distributed architectures with autonomous edge nodes, security and access control are essential [8]. Communication is secured using mTLS and X.509 certificates, ensuring mutual authentication and data protection. Device identity and onboarding are managed via hardware-based TPM and services like Azure DPS or AWS IoT Device Defender, enabling centralized certificate and policy control. Access to services and data is regulated through RBAC, with permissions set at node,

container, or API levels. Additionally, context-aware policies (e.g., geolocation, time, device type) enhance flexible and reliable access control in dynamic, distributed edge environments.

Thus, the technology stack of edge systems covers all levels – from the operating system and container environment to high-level analytics and access control – providing a fully functional and secure implementation of edge computing.

Advantages of using edge computing

Integrating edge computing into production chains significantly reduces information processing delays, increases the autonomy of control nodes, and ensures resilience to network infrastructure failures. This is especially crucial for highly automated enterprises. For example, **Amazon** uses edge computing in its warehouse operations for real-time data processing and inventory management. The company has implemented real-time tracking systems, enabling quick responses to demand changes and optimizing stock levels, thus reducing stockouts and improving inventory accuracy. Additionally, edge computing processes data directly at warehouses, speeding up decision-making and enhancing operational efficiency.

In the aerospace industry, companies also implement edge computing to optimize production and inventory management. For instance, **Boeing** uses edge computing to monitor equipment status and streamline manufacturing processes. This technology improves system autonomy, reduces dependence on cloud services, optimizes inventory management, and enables faster reactions to production changes, all critical for aerospace's high precision and reliability demands.

In aerospace edge infrastructure also provides a reliable platform for local monitoring. For example, an edge node on the assembly line processes data from RFID tags at each workstation. If the assembly sequence is incorrect or a required part is missing, the system can halt the process or alert the central quality system.

Overall, managing inventory at the workshop and warehouse levels is a key application of the edge approach. Using edge devices with connected sensors (RFID, IoT tags, scales, cameras), material movement, status, and availability can be tracked without relying on the cloud.

The integration of edge layers with industrial information systems occurs through API and industrial protocols. ERP systems (e.g., SAP, 1C: ERP), APS (Advanced Planning & Scheduling), and SCADA interact with edge nodes via REST/gRPC interfaces or message brokers (MQTT, AMQP). In practice, edge devices transmit aggregated metadata, preprocessing results, or event signals. SCADA systems use this data for visualization and real-time control, while ERP systems make strategic decisions, including procurement and logistics. Combining edge with APS, for example, allows dynamic production schedule adjustments based on real equipment status and local warehouse material availability.

Thus, using edge computing in production chains not only reduces the load on central resources and improves operational resilience but also enables flexible, locally adapted management scenarios for production, inventory, and quality in the context of digital industrial enterprises.

Conclusion

The evolution of web architectures demonstrates a systemic shift from centralized, rigid solutions to distributed, cloud-based, and ultimately edge-oriented models capable of effectively addressing the challenges of modern industrial digitalization. Integrating edge computing into the information layers of production, inventory, and supply chain management enables not only reduced data processing latency and increased network resilience, but also the implementation of localized analytics and autonomous decision-making. The architectural principles of edge infrastructures – combining containerization, lightweight communication protocols, and secure interaction with enterprise systems – form the technical foundation for high-tech industries, where process continuity and operational precision are critical. Thus, the transition to edge models represents not merely a technological evolution, but a paradigm shift in the design of industrial IT systems.

References

1. Shethiya A.S. Building Scalable and Secure Web Applications Using. NET and Microservices // Academia Nexus Journal. 2025. Vol. 4. No. 1.

2. Asghari A., Sohrabi M.K. Server placement in mobile cloud computing: A comprehensive survey for edge computing, fog computing and cloudlet // Computer Science Review. 2024. Vol. 51. P. 100616.
3. Umarov A. Using technologies to optimize logistics and sales // Universum: economics and jurisprudence: electron. scientific journal. 2025. No. 2(124). P. 51-55.
4. Kumar A., Singh D. Securing IoT devices in edge computing through reinforcement learning // Computers & Security. 2025. Vol. 155. P. 104474.
5. Bolgov S. Automation of business processes using integration platforms and backend technologies // International Research Journal of Modernization in Engineering Technology and Science. 2024. Vol. 6(12). P. 3847-3851.
6. Swain M., Tripathi N., Sethi K. Identifying communication sequence anomalies to detect DoS attacks against MQTT // Computers & Security. 2025. P. 104526.
7. Garifullin R. Development and implementation of high-speed frontend architectures for complex enterprise systems // Cold Science. 2024. No. 12. P. 56-63
8. Bhattacharya T., Peddi A.V., Ponaganti S., Veeramalla S.T. A survey on various security protocols of edge computing // The Journal of Supercomputing. 2025. Vol. 81. No. 1. P. 310.

UDC 004.451.5

APPLICATION OF QUANTUM ALGORITHMS IN BIG DATA ANALYSIS

Khusainov T.Zh.
bachelor's degree, Technical university of Munich (Munich, Germany)

ПРИМЕНЕНИЕ КВАНТОВЫХ АЛГОРИТМОВ В АНАЛИЗЕ БОЛЬШИХ ДАННЫХ

Хусаинов Т.Ж.
бакалавр, Технический университет Мюнхена (Мюнхен, Германия)

Abstract

This article investigates the application of quantum algorithms in the domain of big data analysis, focusing on their theoretical foundations, architectural integration, and sector-specific use cases. The study provides a comparative assessment of classical and quantum approaches to core analytic tasks such as search, optimization, and dimensionality reduction. Key attention is given to hybrid quantum–classical models, implementation challenges, system security, and operational reliability. The article concludes with an overview of current limitations and outlines prospective research directions that can guide the practical deployment of quantum-enhanced analytics in large-scale data environments.

Keywords: quantum algorithms, big data, hybrid computing, optimization, quantum search, security, scalability, quantum analytics.

Аннотация

Статья посвящена применению квантовых алгоритмов в области анализа больших данных, с акцентом на теоретические основы, архитектурные подходы и отраслевые примеры использования. Проведено сравнительное рассмотрение классических и квантовых методов в задачах поиска, оптимизации и снижения размерности. Особое внимание уделяется гибридным архитектурам, проблемам внедрения, вопросам безопасности и надёжности вычислений. В завершение обозначены текущие ограничения и направления дальнейших исследований, направленные на интеграцию квантовых решений в масштабируемые аналитические системы.

Ключевые слова: квантовые алгоритмы, большие данные, гибридные вычисления, оптимизация, квантовый поиск, безопасность, масштабируемость, квантовая аналитика.

Introduction

The exponential growth of data generated across digital ecosystems has rendered traditional computational paradigms increasingly insufficient for efficient large-scale data analysis. As datasets expand in volume, velocity, and variety, classical algorithms face fundamental limitations related to memory bandwidth, processing power, and algorithmic complexity. These challenges have prompted exploration into novel computational models capable of handling such loads with greater efficiency. Among these, quantum computing has emerged as a promising frontier, offering algorithmic speedups for specific classes of problems through principles of superposition and entanglement.

Quantum algorithms, in particular, exhibit notable potential in the domain of big data analytics, where problems often involve searching, clustering, classification, and optimization. Algorithms such as Grover’s search, quantum Fourier transform (QFT), and quantum principal component analysis (QPCA) offer theoretical advantages over their classical counterparts, especially for high-

dimensional datasets. Their applicability ranges from accelerating search operations in unstructured data to enabling quantum-enhanced machine learning models. As quantum hardware continues to evolve, these algorithms are increasingly moving from theoretical constructs toward implementable solutions in hybrid classical–quantum architectures.

The objective of this study is to analyze the potential and limitations of applying quantum algorithms in big data analysis, focusing on their theoretical advantages, current implementations, and practical integration into existing data processing workflows. Special attention is given to the comparative efficiency of quantum versus classical approaches, the suitability of quantum algorithms for various analytic tasks, and the constraints imposed by contemporary quantum hardware. The study aims to provide a structured perspective on how quantum computing can contribute to the transformation of large-scale data analytics.

Main part

Comparative characteristics of quantum and classical algorithms in big data tasks

The application of quantum algorithms in the field of big data analysis has prompted a growing interest in their comparative performance relative to classical approaches. While quantum computing remains in its early stages of physical realization, several algorithms have demonstrated theoretical speedups that could transform the handling of high-dimensional, complex datasets. The selection of algorithmic strategies depends not only on asymptotic performance but also on the structure of the data, the type of task, and the nature of the available quantum hardware [1].

Table 1 provides a comparative overview of several common big data tasks, juxtaposing classical and quantum algorithmic approaches, along with their expected computational advantages. The tasks include unstructured search, matrix operations, dimensionality reduction, combinatorial optimization, and clustering-all of which are core components of modern data analytics pipelines.

Table 1

Comparison of classical and quantum algorithms for key big data tasks

Task	Classical algorithm	Quantum algorithm	Expected speedup
Unstructured search	Linear search (O(n))	Grover's Algorithm (O(√n))	Quadratic
Matrix multiplication	Strassen / Coppersmith-Winograd	Quantum Matrix Multiplication (QMM)	Polylogarithmic (in theory)
Principal component analysis	SVD / Eigen decomposition	Quantum PCA (QPCA)	Exponential (under assumptions)
Optimization (QUBO)	Simulated annealing / Gradient descent	Quantum Approximate Optimization Algorithm (QAOA)	Polynomial (problem-dependent)
Clustering	k-means / DBSCAN	Quantum k-means / VQE clustering	Quadratic

The results indicate that for specific tasks such as unstructured search, Grover’s algorithm offers a proven quadratic speedup, which may be leveraged in contexts such as large database querying or anomaly detection. In optimization problems, particularly those reducible to QUBO (Quadratic Unconstrained Binary Optimization), quantum algorithms like QAOA (Quantum Approximate Optimization Algorithm) provide promising approximations under constrained execution environments.

Dimensionality reduction techniques such as QPCA could significantly outperform classical singular value decomposition (SVD), particularly for massive, sparse matrices. However, these theoretical advantages are conditional on assumptions such as coherent quantum access to the data and sufficiently low noise levels. Moreover, quantum implementations of clustering algorithms

remain in exploratory phases, although initial prototypes (e.g., quantum k-means) show performance improvements in reduced search space exploration.

In summary, while quantum algorithms hold transformative potential in specific computational domains, their integration into big data workflows requires critical consideration of algorithmic maturity, quantum hardware limitations, and data encoding schemes suitable for quantum processing [2].

Architectural models for hybrid quantum–classical data processing

Given the current limitations in quantum hardware, particularly regarding qubit stability and system scale, fully quantum data analysis pipelines remain infeasible for most real-world applications. As a result, hybrid quantum–classical architectures have emerged as a transitional solution, combining the strengths of quantum algorithms with the flexibility and maturity of classical computing. These architectures enable practical experimentation with quantum processing while preserving system-level reliability and scalability.

Hybrid systems typically partition the data analytics workflow into quantum-suitable and classical components. For example, a quantum algorithm may be used for the core computational bottleneck-such as searching or optimization-while data preparation, I/O operations, and final interpretation are managed by classical systems. This division allows organizations to exploit potential quantum speedups without complete migration to quantum infrastructure [3].

Table 2 outlines five architectural patterns commonly adopted in hybrid quantum-classical systems. Each model is characterized by its operational structure, core purpose, and representative use cases across big data domains.

Table 2

Hybrid architectures for quantum–classical big data integration

Architecture type	Description	Use cases
Sequential hybrid	Classical system prepares data and handles output, quantum algorithm performs core computation.	Grover-enhanced search in pre-indexed datasets
Parallel hybrid	Classical and quantum systems work simultaneously on different components of the task.	Hybrid neural network training
Quantum preprocessing	Quantum system performs data encoding or feature transformation before classical analytics.	Quantum-enhanced feature extraction
Quantum postprocessing	Quantum algorithm refines results of prior classical analysis (e.g., optimization).	Post-classical clustering refinement
Federated quantum integration	Multiple quantum nodes integrate into a federated big data pipeline with distributed learning.	Secure collaborative learning across institutions

The sequential hybrid model remains the most accessible, with data preprocessed and postprocessed classically, while the quantum component solves the algorithmic core. This approach is particularly useful for unstructured search and combinatorial problems. In contrast, parallel hybrid models distribute tasks concurrently between quantum and classical systems-such as during hybrid neural network training or reinforcement learning scenarios.

Quantum preprocessing and postprocessing strategies target specific segments of the pipeline to amplify performance, including early-stage feature extraction or late-stage result refinement. Finally, the federated quantum integration model introduces a distributed layer where multiple quantum nodes participate in collaborative analysis-an approach increasingly relevant for secure multi-institution data environments.

These architectural designs reflect a growing maturity in quantum-classical orchestration and indicate viable directions for integrating quantum computing into enterprise-level big data platforms [4].

Challenges and constraints in applying quantum algorithms to big data analysis

Despite their theoretical advantages, quantum algorithms face a range of practical limitations that must be addressed before they can be reliably integrated into big data pipelines. These challenges span across hardware maturity, data representation, algorithm stability, and interoperability with classical systems [5].

One of the most critical constraints is quantum hardware scalability. Current quantum processors are limited in terms of the number of available qubits and the fidelity of quantum gates. For quantum algorithms to outperform classical alternatives on meaningful big data tasks, a significant number of fault-tolerant qubits is required. However, as of now, noisy intermediate-scale quantum (NISQ) devices dominate the landscape, capable of executing only shallow circuits with limited tolerance to decoherence and gate errors.

Another major barrier is quantum data loading, often referred to as the «QRAM bottleneck». For most quantum algorithms to process classical data, that data must first be encoded into a quantum state-an operation that can be costly or even classically inefficient [6]. In the context of big data, where datasets often reach terabyte scale, the question of how to efficiently transform and load structured or unstructured data into quantum memory remains largely unsolved.

Algorithmic fragility is also a concern. Quantum algorithms such as QPCA or QAOA are sensitive to noise, hyperparameter tuning, and circuit depth. Unlike classical algorithms that degrade gracefully with increased noise or data complexity, quantum models often fail catastrophically beyond a certain threshold of uncertainty or decoherence. This raises questions about their robustness and suitability for use in mission-critical analytics systems.

Furthermore, interoperability with classical infrastructure is far from trivial. Big data environments typically rely on established tools like Hadoop, Spark, or cloud-based SQL engines. Embedding quantum computations into these pipelines requires the development of hybrid orchestration layers, data exchange protocols, and quantum-aware middleware-components that are currently in early development or available only as experimental prototypes [7].

In summary, while quantum algorithms present transformative potential for big data analysis, their practical adoption is gated by significant technical and architectural challenges. Addressing these constraints will require advances not only in quantum hardware, but also in algorithm design, software engineering, and systems integration.

Industry-specific use cases of quantum algorithms in big data analytics

The potential of quantum algorithms extends beyond theoretical acceleration, offering practical applications across multiple sectors that depend heavily on large-scale data processing [8]. While real-world deployments remain in their early stages, numerous pilot studies and research collaborations indicate that quantum-enhanced analytics could reshape decision-making, pattern discovery, and optimization in data-intensive industries.

Table 3 presents selected industry use cases in which quantum algorithms are being evaluated or actively researched to augment classical big data workflows [9]. These examples cover critical sectors such as finance, healthcare, telecommunications, energy, and logistics.

Table 3

Industry-specific use cases of quantum algorithms in big data analysis

Industry	Big data task	Quantum approach	Expected impact
Finance	Portfolio optimization, fraud detection	QAOA for portfolio optimization; Grover for anomaly search	Faster decision-making under uncertainty
Healthcare	Genomic data analysis, patient risk profiling	Quantum machine learning for pattern discovery	Improved diagnostic accuracy and personalization
Industry	**Big data task**	**Quantum approach**	**Expected impact**

Telecommunications	Network traffic prediction, anomaly detection	Quantum neural networks for traffic flow modeling	Higher bandwidth efficiency and threat mitigation
Energy	Grid stability analysis, predictive maintenance	Quantum PCA for dimensionality reduction	Enhanced energy distribution and fault detection
Logistics	Route optimization, demand forecasting	Quantum annealing for routing and logistics optimization	Reduced costs and real-time logistics planning

In finance, QAOA are investigated for high-dimensional portfolio management, where classical methods struggle with combinatorial complexity. Likewise, Grover's algorithm has been proposed for real-time fraud detection within unstructured transaction logs.

The healthcare sector benefits from quantum machine learning techniques applied to genomic sequencing and patient risk profiling. Quantum-enhanced pattern recognition may accelerate biomarker discovery and enable more accurate disease classification from massive clinical datasets.

In telecommunications, the use of quantum neural networks has been proposed for modeling traffic flows, predicting network congestion, and detecting anomalous behavior in packet-level data [10]. These approaches aim to improve bandwidth allocation and reduce service interruptions in complex network topologies.

Energy systems rely heavily on forecasting and optimization QPCA is applied to compress grid sensor data while maintaining predictive accuracy. In parallel, quantum algorithms for predictive maintenance help identify fault conditions in turbines and substations before costly failures occur.

Finally, logistics and supply chain operations explore quantum annealing and routing algorithms to optimize delivery routes, schedule fleets, and anticipate fluctuations in demand with greater computational efficiency than classical solvers.

These emerging use cases suggest that quantum algorithms are not merely experimental curiosities but practical tools with transformative potential-especially when embedded within hybrid architectures that complement existing analytics platforms [11].

Future directions and research outlook

As quantum hardware and software ecosystems evolve, new opportunities are emerging for integrating quantum algorithms into scalable big data architectures. One promising direction involves the co-design of quantum algorithms and classical infrastructure to minimize communication overhead and leverage specialized hardware accelerators. Future systems are expected to blend quantum co-processors with edge computing nodes and high-performance clusters, enabling real-time quantum-enhanced analytics in distributed environments.

Another active area of research focuses on quantum data representation and encoding strategies. Efficient methods for mapping classical datasets into quantum states-without incurring exponential costs-remain a prerequisite for any practical deployment. Techniques such as amplitude encoding, basis embedding, and variational circuits are currently being refined to support this transition, with particular emphasis on sparse and high-dimensional datasets common in industrial settings.

Moreover, the development of standardized benchmarks for performance evaluation is essential. While theoretical speedups are widely cited, empirical validation on near-term hardware is limited [12]. Establishing common metrics and datasets for comparing classical and quantum models across diverse analytics tasks will improve reproducibility and foster trust in experimental outcomes.

Privacy-preserving computation is also gaining traction, especially in sectors where sensitive data prohibits centralized processing. Quantum-secured federated learning, homomorphic encryption integration, and post-quantum cryptographic resilience are likely to converge with data analytics pipelines, creating hybrid protocols that balance performance and confidentiality.

Finally, interdisciplinary collaboration will be critical to realize the full potential of quantum data analysis. Researchers in quantum information science, distributed systems, software engineering, and applied machine learning must work in tandem to bridge theoretical advances with engineering

feasibility. As quantum computing moves from lab-scale prototypes to enterprise adoption, these collaborative frameworks will guide the responsible and effective deployment of quantum-enhanced big data solutions.

Security and reliability in quantum big data pipelines

As quantum computing capabilities expand, ensuring the security and reliability of quantum-enhanced big data systems becomes a central concern. The unique characteristics of quantum algorithms-such as reversibility, entanglement-based correlations, and probabilistic outputs-introduce new classes of vulnerabilities that must be addressed through both architectural safeguards and algorithmic hardening.

A key issue lies in the opacity of quantum model behavior. Many quantum algorithms produce outcomes that are statistically sampled from a probability distribution, making deterministic interpretation and reproducibility more difficult. In critical applications such as fraud detection or medical diagnostics, the inability to consistently trace the reasoning behind a quantum decision may undermine trust and compliance with regulatory frameworks.

Another challenge is model exposure in hybrid pipelines. Quantum circuits, especially those deployed via cloud-based quantum platforms, may become targets of reverse engineering or extraction attacks. Just as classical models can be cloned or adversarially probed through APIs, quantum models are theoretically susceptible to similar threats-particularly when measurement data or circuit structure is leaked. This risk is amplified in distributed pipelines where quantum components are repeatedly queried or invoked via orchestration frameworks.

From an infrastructure perspective, error propagation in quantum computation poses significant reliability risks. Unlike classical faults, quantum errors can cascade non-linearly due to entangled states and superposition, potentially contaminating results across dependent subsystems. Without robust error correction, which remains experimentally challenging, systems may produce degraded analytics outputs without immediate detection.

To mitigate these concerns, several defensive strategies are emerging. Quantum circuit obfuscation, encrypted execution environments, and differentially private measurement protocols offer partial protection against model misuse. In parallel, hardware-level solutions-such as isolated quantum memory and authenticated access control-are being developed to secure quantum processing units in multi-tenant cloud infrastructures [13].

In terms of reliability, efforts are underway to design redundant hybrid configurations, in which classical subroutines validate or cross-check the outputs of quantum modules. This layered architecture not only provides failover capabilities but also introduces audit trails and confidence scoring mechanisms, which are essential in risk-sensitive analytics pipelines.

Ultimately, as quantum components are introduced into big data environments, security and reliability must be treated as first-class architectural principles-embedded into every layer of the analytic workflow, from data ingestion to inference.

Conclusion

The application of quantum algorithms to big data analysis marks a critical juncture in the evolution of computational science. By leveraging the unique capabilities of quantum systems-such as superposition, entanglement, and probabilistic inference-it becomes possible to address analytic tasks that exceed the practical limits of classical computing. Search, optimization, dimensionality reduction, and pattern recognition are among the domains where quantum methods demonstrate theoretical speedups and architectural advantages.

This study has examined the comparative performance of classical and quantum approaches across core big data tasks, proposed practical hybrid architectures for implementation, and analyzed current barriers including data loading, noise sensitivity, and interoperability. Furthermore, it has outlined potential industry use cases, emphasized security and reliability concerns, and highlighted areas for continued research.

While full-scale adoption of quantum-enhanced analytics remains dependent on further advancements in hardware stability, data encoding schemes, and ecosystem maturity, early integration into hybrid workflows has already begun. Quantum algorithms should no longer be regarded solely

as future theoretical constructs, but as emerging tools that-when properly applied-can contribute to the scalability, efficiency, and intelligence of modern big data systems.

References

1. Ramírez J.G.C. Advanced quantum algorithms for big data clustering and high-dimensional classification // Journal of Advanced Computing Systems. 2024. Vol. 4. No. 6.
2. Anand S. Quantum Computing for Large-Scale Healthcare Data Processing: Potential and Challenges // International Journal of Emerging Trends in Computer Science and Information Technology. 2023. Vol. 4. No. 4. P. 49-59.
3. Bova F., Goldfarb A., Melko R.G. Commercial applications of quantum computing // EPJ quantum technology. 2021. Vol. 8. No. 1. P. 2.
4. Nurhayaty M., Manullang J. Development of Quantum Algorithms for Neural Network Optimization in Big Data Analysis // Journal ICT: Information and Communication Technologies. 2023. Vol. 14. No. 2. P. 31-35.
5. Deng L., Wan L., Guo J. Research on security anomaly detection for big data platforms based on quantum optimization clustering // Mathematical Problems in Engineering. 2022. No. 1. P. 4805035.
6. Ur Rasool R., Ahmad H.F., Rafique W., Qayyum A., Qadir J., Anwar Z. Quantum computing for healthcare: A review // Future Internet. 2023. Vol. 15. No. 3. P. 94.
7. Sood S.K. Quantum computing review: A decade of research // IEEE Transactions on Engineering Management. 2023. Vol. 71. P. 6662-6676.
8. Osaba E., Villar-Rodriguez E., Oregi I. A systematic literature review of quantum computing for routing problems // IEEE Access. 2022. Vol. 10. P. 55805-55817.
9. Khan A.A., Ahmad A., Waseem M., Liang P., Fahmideh M., Mikkonen T., Abrahamsson P. Software architecture for quantum computing systems - A systematic review // Journal of Systems and Software. 2023. Vol. 201. P. 111682.
10. Yang Z., Zolanvari M., Jain R. A survey of important issues in quantum computing and communications // IEEE Communications Surveys & Tutorials. 2023. Vol. 25. No. 2. P. 1059-1094.
11. Batra K., Zorn K.M., Foil D.H., Minerali E., Gawriljuk V.O., Lane T.R., Ekins S. Quantum machine learning algorithms for drug discovery applications // Journal of chemical information and modeling. 2021. Vol. 61. No. 6. P. 2641-2647.
12. Cerezo M., Verdon G., Huang H.Y., Cincio L., Coles P.J. Challenges and opportunities in quantum machine learning // Nature computational science. 2022. Vol. 2. No. 9. P. 567-576.
13. Berry D.W., Su Y., Gyurik C., King R., Basso J., Barba A.D.T., Babbush R. Analyzing prospects for quantum advantage in topological data analysis // PRX Quantum. 2024. Vol. 5. No. 1. P. 010319.

УДК 004.056

ЦИФРОВАЯ ИДЕНТИЧНОСТЬ И РАСПРЕДЕЛЁННЫЕ РЕЕСТРЫ В E-GOVERNMENT СИСТЕМАХ

Ахметшин В.Р.
специалист, Кубанский государственный технологический университет (Краснодар, Россия)

Тумашев А.Г.
специалист, Кубанский государственный технологический университет (Краснодар, Россия)

DIGITAL IDENTITY AND DISTRIBUTED LEDGERS IN E-GOVERNMENT SYSTEMS

Akhmetshin V.R.
specialist degree, Kuban state technological university (Krasnodar, Russia)

Tumashev A.G.
specialist degree, Kuban state technological university (Krasnodar, Russia)

Аннотация

Цифровая идентичность, основанная на технологиях распределённых реестров (DLT), становится фундаментом для построения доверенных e-government-систем нового поколения. Эффективное внедрение таких решений требует учёта архитектурных, нормативных и инфраструктурных факторов. Проведён сравнительный анализ зрелости внедрения цифровых удостоверений личности в международной практике, а также технических характеристик ключевых DLT-платформ. Обоснована значимость самоуправляемой идентичности (SSI) как механизма повышения приватности и пользовательского контроля. Предложен подход к выбору технологической архитектуры, учитывающий баланс между производительностью, нормативной совместимостью и требованиями к масштабируемости. Полученные результаты могут быть использованы при разработке национальных стратегий цифровой трансформации.

Ключевые слова: цифровая идентичность, DLT, SSI, e-government, блокчейн, масштабируемость, архитектура, приватность.

Abstract

Digital identity built on distributed ledger technologies (DLT) is emerging as a foundational component of next-generation e-government infrastructures. Successful implementation depends on a combination of architectural design, regulatory frameworks, and technical interoperability. This paper presents a comparative assessment of global maturity levels in DLT-based identity systems and evaluates core platform characteristics. The relevance of self-sovereign identity (SSI) is emphasized as a means to enhance user control and privacy. A methodological approach is proposed for selecting technological architectures that align performance, scalability, and compliance objectives. The findings provide practical guidance for national digital transformation strategies and public sector implementation.

Keywords: digital identity, DLT, SSI, e-government, blockchain, architecture, scalability, privacy.

Введение

Современные цифровые трансформации в государственном управлении обусловлены необходимостью повышения прозрачности, эффективности и доверия в отношениях между гражданами, бизнесом и государством. Одним из ключевых элементов этих преобразований становится развитие электронного правительства (e-government), ориентированного на автоматизацию административных процессов, цифровизацию сервисов и интеграцию распределённых информационных систем.

Важнейшим компонентом e-government-инфраструктуры выступает цифровая идентичность - совокупность уникальных цифровых признаков, позволяющих однозначно идентифицировать субъекта в виртуальной среде. Безопасность, устойчивость и управляемость цифровых идентичностей напрямую влияют на качество предоставляемых государственных услуг, включая регистрацию граждан, доступ к медицинским и образовательным системам, участие в выборах и распределение социальных выплат. В условиях растущих угроз приватности и фальсификации данных особое значение приобретают подходы, основанные на распределённых реестрах (distributed ledgers), в том числе на технологии блокчейн.

Цель данной статьи - проанализировать потенциал интеграции цифровой идентичности с распределёнными реестрами в рамках e-government-систем. Рассматриваются технические, правовые и организационные аспекты внедрения распределённых идентификационных моделей, оцениваются преимущества и ограничения подхода, а также предлагается концептуальная модель архитектуры государственной системы цифровой идентификации, основанной на децентрализованных технологиях.

Основная часть. Эволюция цифровой идентичности в контексте электронного государства

Развитие электронного правительства сопровождалось последовательной трансформацией моделей идентификации пользователей. На ранних этапах цифровизации основное внимание уделялось централизованным системам аутентификации, в которых государственные органы или доверенные провайдеры удостоверяли личность пользователя на основе традиционных документов и закрытых реестров [1]. Такие подходы обеспечивали базовый уровень надёжности, однако оставались уязвимыми к утечкам данных, техническим сбоям и ограничениям масштабируемости.

С переходом к более зрелым e-government-платформам возникла необходимость в создании самостоятельных и гибко управляемых цифровых идентичностей, которые могли бы использоваться повторно в различных административных и межведомственных сценариях. Это привело к распространению моделей федеративной идентификации, в которых несколько ведомств или сервисов признают одну и ту же цифровую сущность. Однако и такие решения сохраняют зависимость от центральных регистраторов и не устраняют риски, связанные с единым центром отказа.

Современные концепции цифровой идентичности всё чаще ориентированы на децентрализованные и пользовательско-ориентированные модели. Примером такой парадигмы выступает концепция самоуправляемой идентичности (self-sovereign identity, SSI), согласно которой субъект сам контролирует доступ к своим данным, а их верификация осуществляется через распределённые реестры с использованием криптографических механизмов доверия. В рамках SSI доверие строится не на инфраструктуре централизованных удостоверяющих центров, а на сетевом консенсусе и цифровых подписях, записываемых в неизменяемые блоки данных.

Таким образом, эволюция цифровой идентичности в e-government-среде демонстрирует переход от централизованного управления к распределённым и суверенным моделям, где безопасность, приватность и прозрачность определяются архитектурными особенностями цифровой экосистемы, а не исключительно институциональным доверием.

Архитектурные модели цифровой идентичности в e-government-среде

Выбор архитектуры цифровой идентичности в системах электронного правительства предопределяет не только технологическую реализацию, но и нормативные аспекты взаимодействия между государством, гражданином и сторонними сервисами [2]. Современные подходы к проектированию таких систем можно условно разделить на три категории: централизованная, федеративная и децентрализованная (на основе распределённых реестров).

Централизованная архитектура предполагает наличие единого удостоверяющего органа, отвечающего за регистрацию, хранение и верификацию идентификационных данных. Этот подход широко применялся на ранних этапах цифровизации, но характеризуется высокой уязвимостью к атакам, а также рисками злоупотребления полномочиями.

Федеративные модели позволяют объединять несколько поставщиков удостоверяющих услуг в рамках доверенной сети. Верификация может осуществляться с использованием протоколов, таких как SAML или OAuth2, при этом данные пользователя могут циркулировать между различными ведомствами. Несмотря на более высокую гибкость, федеративные решения всё ещё полагаются на центральную инфраструктуру и требуют строгой координации между участниками.

Децентрализованные архитектуры, реализуемые на базе распределённых реестров, принципиально изменяют модель доверия. В системе SSI каждый субъект (гражданин, организация, государственное учреждение) может создавать и управлять собственными идентификаторами, а проверки подлинности осуществляются через цифровые доказательства и записи в блокчейне. Такие системы повышают прозрачность и устойчивость, снижая зависимость от централизованных доверенных сторон.

На рисунке 1 представлена сравнительная схема трёх архитектур цифровой идентичности, применяемых в e-government-системах.

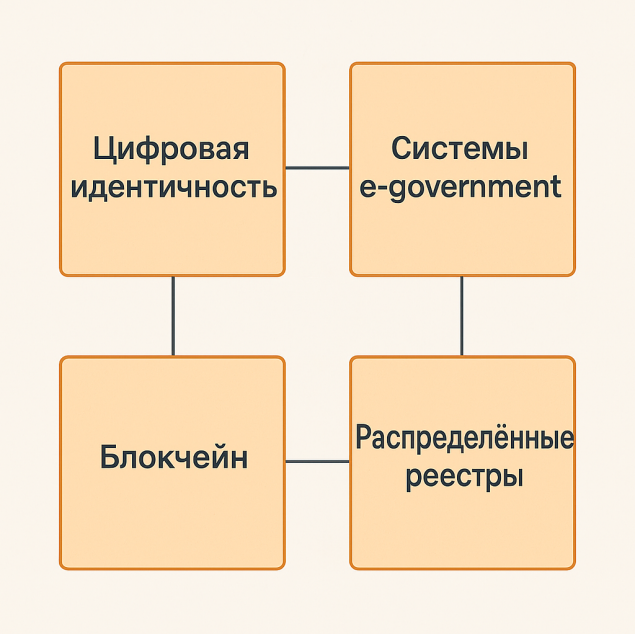


Рисунок 1. Сравнение архитектур цифровой идентичности: централизованная, федеративная и децентрализованная модель

Сравнительный анализ архитектур цифровой идентичности, представленных на рисунке, демонстрирует, что централизованные модели, несмотря на простоту реализации, уступают по уровню отказоустойчивости и контролю со стороны пользователей. Федеративные решения обеспечивают более высокий уровень взаимодействия между ведомствами, но сохраняют зависимость от доверенных посредников. Наиболее перспективной в контексте цифрового суверенитета граждан и устойчивости инфраструктуры представляется децентрализованная модель, реализуемая с использованием распределённых реестров. Она обеспечивает наибольшую степень прозрачности, масштабируемости и криптографической надёжности, что особенно актуально для современных e-government-систем [3].

Сравнительный анализ решений цифровой идентификации на базе DLT

Внедрение технологий распределённых реестров (Distributed Ledger Technology, DLT) в государственные идентификационные системы сопровождается ростом интереса к конкретным технологическим платформам. На сегодняшний день существует несколько зрелых решений, адаптированных под нужды e-government: Hyperledger Indy, Ethereum, Sovrin, Quorum и другие. Каждое из них обладает специфическими характеристиками, влияющими на уровень доверия, масштабируемость, приватность и юридическую совместимость.

В таблице 1 приведено подробное сравнение ключевых DLT-платформ, применяемых для построения систем цифровой идентичности, включая их технологические характеристики, уровень приватности, совместимость с государственными системами и другие важные параметры.

Таблица 1

Сравнение ключевых DLT-платформ для построения систем цифровой идентичности: особенности и ограничения

Платформа	**Тип реестра и поддержка SSI**	**Приватность и масштабируемость**	**Применимость в e-government**
Hyperledger indy	Разрешённый, полная поддержка самоуправляемой идентичности	Высокий уровень приватности, средняя масштабируемость	Рекомендуется для интеграции с госреестрами и удостоверяющими центрами
Ethereum	Публичный, ограниченная поддержка концепции идентичности	Низкая приватность, высокая масштабируемость	Ограниченная применимость, требует доработки под госстандарты
Sovrin	Публично-разрешённый, полная поддержка самоуправляемой идентичности	Высокая приватность, средняя масштабируемость	Подходит для национальных проектов, интегрируется с DID и VC
Quorum	Разрешённый, частичная поддержка пользовательского контроля	Средняя приватность, высокая масштабируемость	Может использоваться в ведомственных системах с настройкой приватности
Corda	Разрешённый, ограниченная поддержка через плагины и настройки	Высокая приватность, высокая масштабируемость	Частично применима, требует адаптации для идентификационных задач

Анализ представленных в таблице платформ показывает, что наибольший потенциал для внедрения в государственные идентификационные системы демонстрируют решения Hyperledger Indy и Sovrin, ориентированные на поддержку концепции самоуправляемой идентичности и обеспечение высокого уровня приватности. Платформы общего назначения, такие как Ethereum и Quorum, хотя и обладают высокой масштабируемостью, в базовой конфигурации недостаточно адаптированы к требованиям защиты персональных данных и нормативной совместимости. Решения на базе Corda могут быть применимы в межведомственных сценариях, но требуют доработки механизмов идентификации. Следовательно, выбор технологии должен базироваться на комплексной оценке приватности, масштабируемости и соответствия регуляторным требованиям конкретной юрисдикции.

Важным аспектом выбора DLT-платформы также становится уровень зрелости экосистемы, наличие поддержки со стороны разработчиков и соответствие национальным стандартам информационной безопасности [4]. Так, Hyperledger Indy уже адаптирован для

интеграции с национальными идентификационными системами в ряде стран, включая пилотные проекты в Канаде и Индии, что делает его привлекательным для масштабируемых решений в сфере государственного управления.

Кроме того, при проектировании идентификационных систем на базе DLT особое внимание должно уделяться интероперабельности. Возможность обмена данными между платформами и ведомствами без потери доверия критически важна для построения единой цифровой идентичности, пригодной для использования в различных сферах - от налоговых сервисов до электронного здравоохранения. Такие механизмы, как DID (Decentralized Identifier) и VC (Verifiable Credentials), становятся ключевыми стандартами, способствующими совместимости между системами, независимо от выбранной базовой инфраструктуры.

Наконец, не менее значимым фактором является гибкость лицензирования и модели управления платформой. Платформы с открытым исходным кодом, управляемые консорциумами или фондами (например, Hyperledger Foundation), обеспечивают большую прозрачность и возможность адаптации к локальным требованиям. В то время как проприетарные решения могут ограничивать развитие за счёт зависимости от конкретного вендора и стоимости владения.

Уровни зрелости внедрения цифровой идентичности с DLT в международной практике

Внедрение цифровой идентичности с использованием распределённых реестров в e-government-системах находится на разных стадиях зрелости в зависимости от политико-экономического контекста, нормативного регулирования и цифровой зрелости страны. По состоянию на текущий этап развития можно выделить несколько уровней зрелости, отражающих глубину интеграции DLT в государственную инфраструктуру:

Начальный уровень - страны, исследующие возможности применения блокчейн-технологий, но не приступившие к пилотным внедрениям (например, Аргентина, Казахстан).

Пилотный уровень - реализуются ограниченные проекты в отдельных ведомствах или регионах, как правило, без масштабной нормативной базы (например, Индия, Украина).

Интеграционный уровень - платформа цифровой идентичности на базе DLT включена в e-government-экосистему, охватывая несколько сфер: здравоохранение, налоги, голосование (например, Эстония, Канада).

Стабильный уровень - устойчиво работающая, проверенная временем и регулированием архитектура, способная масштабироваться и соответствовать международным стандартам (например, Южная Корея, Сингапур) [5].

Правовые и этические аспекты применения DLT в системах цифровой идентичности

Внедрение цифровой идентичности с использованием распределённых реестров в государственные информационные системы требует тщательного учета правовых, этических и регуляторных аспектов. Особую актуальность эти вопросы приобретают в контексте хранения, обработки и верификации персональных данных, а также в условиях трансграничного взаимодействия.

С точки зрения законодательства, основным ориентиром для большинства стран остаются положения GDPR (General Data Protection Regulation), а также национальные законы о защите персональных данных. Использование неизменяемых и публичных блокчейнов может противоречить праву на удаление информации («право быть забытым»), что создаёт противоречие между принципами децентрализации и нормативными требованиями [6].

Одним из путей решения данной проблемы становится применение off-chain-хранилищ или хешированных ссылок, где сами персональные данные не записываются в реестр, а используются ссылки на зашифрованные внешние хранилища. Это позволяет сохранить преимущества блокчейн-модели (прозрачность и неизменяемость) без нарушения конфиденциальности.

Кроме правового регулирования, значимую роль играют этические вопросы, связанные с цифровым неравенством и возможным навязыванием технологий уязвимым группам

населения. Обязательное использование цифровой идентичности на основе DLT может создавать барьеры для лиц с ограниченным доступом к цифровым устройствам или интернету.

Не менее важным является обеспечение институциональной подотчётности. Применение самоуправляемой цифровой идентичности требует наличия доверенной инфраструктуры: агентств по выпуску идентификаторов, органов аудита, а также механизмов разрешения споров. Без этих элементов любые децентрализованные архитектуры рискуют потерять легитимность в глазах пользователей и регуляторов.

Таким образом, разработка цифровой идентичности на базе DLT должна основываться не только на технологической состоятельности, но и на чётком соблюдении правовых норм, принципов справедливости и социального баланса. Это требует междисциплинарного подхода с участием технологов, юристов, представителей государства и гражданского общества.

Модели доверия и распределение ответственности в DLT-системах цифровой идентичности

В архитектуре цифровой идентичности на основе распределённых реестров важнейшую роль играет модель распределения доверия и ответственности между участниками системы. В отличие от централизованных решений, где вся логика и ответственность сосредоточены в одном органе, распределённые модели предполагают координацию между несколькими ролями:

Issuer (Эмитент) - организация, уполномоченная на выпуск идентификаторов или аттестатов (например, налоговая служба, университет).

Holder (Держатель) - субъект идентичности (гражданин, юридическое лицо), хранящий свои идентификационные данные и управляющий доступом к ним.

Verifier (Проверяющий) - сторона, запрашивающая подтверждение тех или иных атрибутов личности (например, работодатель или банк).

Registry node (Узел реестра) - участник сети, поддерживающий инфраструктуру DLT, обеспечивая неизменяемость и доступность записей.

Governance authority (Регулятор) - надзорный орган, устанавливающий технические и юридические рамки функционирования всей системы.

На рисунке 2 представлена схема распределения этих ролей и потока доверия между ними.

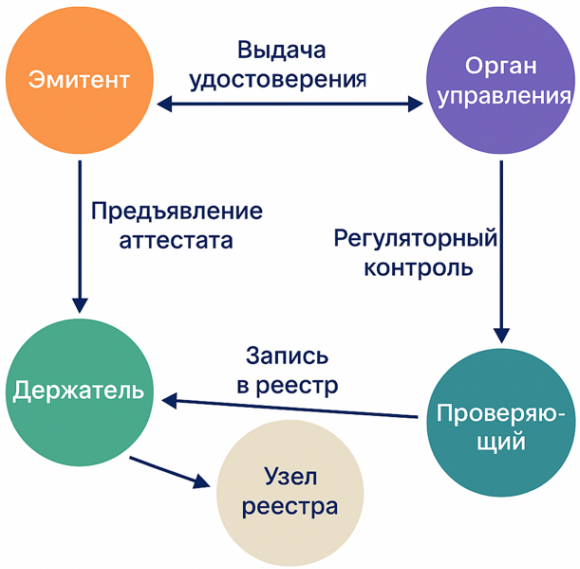


Рисунок 2. Распределение ролей и потоков данных в системе цифровой идентичности на основе DLT

Представленная схема демонстрирует ключевое распределение ролей и взаимодействий между участниками децентрализованной системы цифровой идентичности. В отличие от централизованных моделей, архитектура на основе DLT предусматривает передачу ответственности от единого оператора к множеству доверенных сторон: эмитентов, держателей и проверяющих [7]. Такая модель повышает прозрачность, масштабируемость и

устойчивость системы, однако требует строгой координации между ролями и наличия регулирующей инфраструктуры. Потоки данных между участниками организованы на принципах избирательного доступа и криптографической верификации, что обеспечивает баланс между приватностью и достоверностью.

Технологические ограничения и проблемы масштабируемости DLT в системах цифровой идентичности

Несмотря на широкое обсуждение преимуществ DLT в контексте цифровой идентичности, их внедрение в государственные системы сталкивается с рядом технологических ограничений, способных существенно замедлить или усложнить масштабное применение.

Одним из ключевых вызовов является низкая пропускная способность и ограниченная масштабируемость большинства DLT-платформ. При большом количестве транзакций - например, при массовом обращении граждан к электронным сервисам - система может испытывать задержки или перегрузку. Особенно это актуально для публичных блокчейнов, где все узлы должны достичь консенсуса, что увеличивает время обработки.

Другой критический аспект - энергопотребление и производительность. Хотя некоторые платформы (например, Tezos, Algorand) уже реализуют энергоэффективные протоколы, большая часть существующих решений всё ещё базируется на ресурсоёмких алгоритмах, не соответствующих принципам устойчивого развития и «зелёных» ИТ [8].

Также остаётся нерешённой проблема интероперабельности между различными DLT-платформами. В условиях, когда государственная система должна взаимодействовать с коммерческими, банковскими и международными сервисами, отсутствие стандартов затрудняет обмен данными, а также усложняет аудит и миграцию между платформами.

Дополнительные риски возникают в связи с обновлением и модификацией данных, что критично для цифровой идентичности. Технология блокчейн по своей природе неизменяема, а это затрудняет реализацию механизмов отзыва удостоверений, внесения исправлений или удаления данных в соответствии с нормативными требованиями.

Наконец, важно учитывать сложность интеграции DLT-решений с устаревшими государственными ИС, где отсутствует единый протокол или интерфейс взаимодействия. Это требует вложений в адаптационные слои, шлюзы и API, что увеличивает стоимость и сроки внедрения.

Таким образом, внедрение DLT в e-government-системы цифровой идентичности требует не только нормативной и организационной готовности, но и серьёзной технической адаптации, ориентированной на устойчивость, совместимость и производительность.

Сравнение DLT-платформ по критериям применимости в e-government

Выбор технологической платформы для реализации цифровой идентичности в системах электронного правительства определяется набором ключевых критериев, от которых зависит не только производительность, но и юридическая совместимость, устойчивость и масштабируемость архитектуры. Среди таких критериев выделяются:

- Производительность и пропускная способность - способность системы обрабатывать большое количество транзакций в секунду без снижения стабильности.
- Энергоэффективность - важный показатель устойчивости и экосоциальной совместимости технологии.
- Интероперабельность - возможность взаимодействия с другими платформами и стандартами цифровой идентичности.
- Поддержка SSI (Self-Sovereign Identity) - наличие встроенных механизмов управления удостоверениями со стороны пользователей.
- Гибкость и адаптируемость - возможность настройки архитектуры под нормативные и организационные особенности разных стран [9].

Анализ доступных DLT-платформ показывает, что ни одна из них не обладает абсолютным превосходством по всем критериям. Ethereum, как одна из наиболее распространённых публичных блокчейн-платформ, обеспечивает высокую совместимость с

децентрализованными приложениями, однако страдает от ограниченной масштабируемости и высокой энергозатратности в своей классической реализации (до перехода на алгоритм консенсуса Proof-of-Stake). При этом адаптация Ethereum для нужд цифровой идентичности требует внедрения дополнительных протоколов и надстроек.

Hyperledger Indy и связанная с ней экосистема Sovrin изначально ориентированы на реализацию концепции самоуправляемой идентичности (SSI). Эти платформы демонстрируют высокий уровень соответствия требованиям приватности, управления доступом и децентрализованной верификации, но уступают в гибкости и интеграционной совместимости с другими системами, а также требуют более сложной настройки инфраструктуры.

Quorum и Corda, разработанные для корпоративных и межведомственных сценариев, показывают хорошие результаты в части масштабируемости, управляемости и регуляторной совместимости. Тем не менее, в базовой конфигурации эти решения не содержат встроенной поддержки SSI и требуют дополнительной настройки для реализации пользовательского контроля над удостоверениями.

Таким образом, выбор DLT-платформы должен осуществляться не только на основании технических характеристик, но и с учётом требований конкретной юрисдикции, уровня зрелости ИТ-инфраструктуры, сценариев использования и политико-правовой среды [10]. В некоторых случаях оправдано применение гибридных моделей, где один уровень системы функционирует на основе публичного блокчейна, а другой - на приватной или разрешённой сети, что позволяет достичь компромисса между открытостью, контролем и безопасностью.

Заключение

Развитие цифровой идентичности на основе DLT представляет собой важнейшее направление цифровой трансформации государственного управления. Такие технологии позволяют повысить уровень доверия, прозрачности и пользовательского контроля в системах идентификации, что особенно актуально в условиях растущих требований к защите персональных данных, обеспечению киберустойчивости и международной совместимости.

В статье был проведён анализ архитектурных моделей, технологических платформ и практик внедрения цифровой идентичности в различных странах. Показано, что наибольшую зрелость в этом направлении демонстрируют государства с развитой цифровой инфраструктурой, нормативной базой и стратегией внедрения самоуправляемых удостоверений.

Ключевые технологические ограничения - масштабируемость, интероперабельность, энергоэффективность и сложность интеграции - остаются значимыми барьерами. Их преодоление возможно через применение гибридных архитектур, развитие стандартов и институциональную поддержку со стороны государства. Важно подчеркнуть, что универсального решения не существует: каждая страна должна разрабатывать собственную модель цифровой идентичности с учётом локального контекста, при этом опираясь на международный опыт и открытые технологии.

В перспективе именно DLT может стать основой для построения по-настоящему доверенных, распределённых и самоуправляемых цифровых удостоверений личности, способных трансформировать не только e-government, но и всю экосистему цифровых услуг.

References

1. Elisa N., Yang L., Chao F., Cao Y. A framework of blockchain-based secure and privacy-preserving E-government system // Wireless networks. 2023. Vol. 29. No. 3. P. 1005-1015.
2. Sung C.S., Park J.Y. Understanding of blockchain-based identity management system adoption in the public sector // Journal of Enterprise Information Management. 2021. Vol. 34. No. 5. P. 1481-1505.
3. Ranjith Kumar M.V., Bhalaji N. Blockchain based chameleon hashing technique for privacy preservation in E-governance system // Wireless Personal Communications. 2021. Vol. 117. No. 2. P. 987-1006.

4. Mustafa G., Rafiq W., Jhamat N., Arshad Z., Rana F.A. Blockchain-based governance models in e-government: a comprehensive framework for legal, technical, ethical and security considerations // International Journal of Law and Management. 2025. Vol. 67. No. 1. P. 37-55.
5. Kuperberg M., Kemper S., Durak C. Blockchain usage for government-issued electronic IDs: A survey // International Conference on Advanced Information Systems Engineering. Cham: Springer International Publishing. 2019. P. 155-167.
6. Elisa N., Yang L., Chao F., Naik N., Boongoen T. A secure and privacy-preserving e-government framework using blockchain and artificial immunity // IEEE Access. 2023. Vol. 11. P. 8773-8789.
7. Al-Ameri H.H., Ayvaz S. A Blockchain-Based Secure Mutual Authentication System for E-Government Services // 2023 3rd International Scientific Conference of Engineering Sciences (ISCES). IEEE. 2023. P. 19-24.
8. Datta A. Blockchain enabled digital government and public sector services: A survey // Blockchain and the Public Sector: Theories, Reforms, and Case Studies. 2021. P. 175-195.
9. Gao Y., Pan Q., Liu Y., Lin H., Chen Y., Wen Q. The notarial office in E-government: a blockchain-based solution // IEEE Access. 2021. Vol. 9. P. 44411-44425.
10. Fajar A., Trialih R., Ramlan F.W. A Decentralized File Storage for Effective E-Government // Indonesia Post-Pandemic Outlook: Environment and Technology Role for Indonesia Development. 2022. P. 279.

UDC 004.42

DEVELOPMENT OF VISUAL EDITORS FOR DIGITAL MEDIA: ARCHITECTURE, WEBSITE INTEGRATION, AND ADVERTISING POTENTIAL OF INTERACTIVE CONTENT

Andreev G.
bachelor's degree, Moscow Polytechnic University (Moscow, Russia)

РАЗРАБОТКА ВИЗУАЛЬНЫХ РЕДАКТОРОВ ДЛЯ ЦИФРОВЫХ СМИ: АРХИТЕКТУРА, ИНТЕГРАЦИЯ С САЙТАМИ И РЕКЛАМНЫЙ ПОТЕНЦИАЛ ИНТЕРАКТИВНОГО КОНТЕНТА

Андреев Г.А.
бакалавр, Московский политехнический университет (Москва, Россия)

Abstract

The article analyzes the engineering and architectural foundations underpinning the development of visual editors for digital media. It examines principles of modularity, extensibility, and separation of concerns, which enable the creation of interactive content without programmer involvement. It is emphasized that such editors support the automation of multimedia content production, facilitate integration with websites via API, and ensure secure isolation of executable code. The article also examines monetization strategies that include integrating advertising formats into visual media and assesses the impact of interactive features on audience engagement metrics. It is emphasized that utilizing visual editors aids in streamlining resource usage and fostering a technology-resistant framework for digital media creation.

Keywords: visual editor, digital journalism, interactive content, user engagement, advertising, marketing.

Аннотация

В статье анализируются инженерные и архитектурные решения, лежащие в основе разработки визуальных редакторов для цифровых медиа. Исследуются принципы модульности, расширяемости и разделения ответственности, позволяющие обеспечить работу с интерактивным контентом без участия программистов. Подчёркивается, что такие редакторы позволяют автоматизировать процесс создания мультимедийных публикаций, интегрироваться с сайтами через API и обеспечивать безопасную изоляцию исполняемого кода. Кроме того, в статье рассматриваются модели монетизации, предполагающие интеграцию рекламных форматов в структуру визуальных публикаций, а также проводится анализ влияния интерактивных механизмов на поведенческие метрики аудитории. Подчеркивается, что использование визуальных редакторов способствует оптимизации ресурсозатрат и формированию технологически устойчивой инфраструктуры цифрового медиапроизводства.

Ключевые слова: визуальный редактор, цифровая журналистика, интерактивный контент, вовлеченность пользователей, реклама, маркетинг.

Introduction

The swift progress of digital technologies, along with changing trends in media consumption, has caused a change in the formats employed for delivering journalistic content. Modern viewers –

especially younger ones – show a steady inclination for visually appealing, interactive, and animated content that reveals notably greater engagement than conventional text-focused publications. The creation of such content often necessitates the participation of expert technical teams, rendering the process labor-intensive, time-consuming, and expensive regarding both development and continuous upkeep.

In response to these challenges, there is growing interest in tools that democratize the creation of visual content. One such solution is the integration of embedded visual editors within the editorial systems of digital media platforms. These editors enable designers and editorial staff to independently assemble full-fledged multimedia articles incorporating animations, interactive components, and visual narratives – without the need for developer involvement. The rendering and functionality of these visual publications are managed via a universal client-side script delivered from the editor vendor's cloud infrastructure. This architectural approach facilitates centralized control over presentation logic, ensures backward compatibility with previously published materials, and allows for rapid deployment of new features without requiring modifications to individual articles.

The aim of this study is to analyze the engineering and architectural foundations underlying the development of visual editors for digital media, to explore their integration within existing website infrastructures, and to examine their role in enabling new models of monetization through interactive advertising content.

Main part. The transformation of content formats in digital journalism

The evolution of visual and interactive genres in online journalism has introduced radical changes to forms of presenting information, away from the traditional text and static images towards dynamic, multimedia, and interactive journalism. Although text accompanied by images continued to be the primary format in the initial phases of digitalization, multimedia longreads have become more prominent since the mid-2010s. They consist of images, infographics, integrated videos, and animations, all contained in a unified visual story.

As learned, the population on the internet grew to over 5,5 billion during the period 2005-2024. This rapid increase not only heightened access to online media, but it totally revolutionized expectations on the part of audiences regarding the shape and scope of information content (fig. 1).

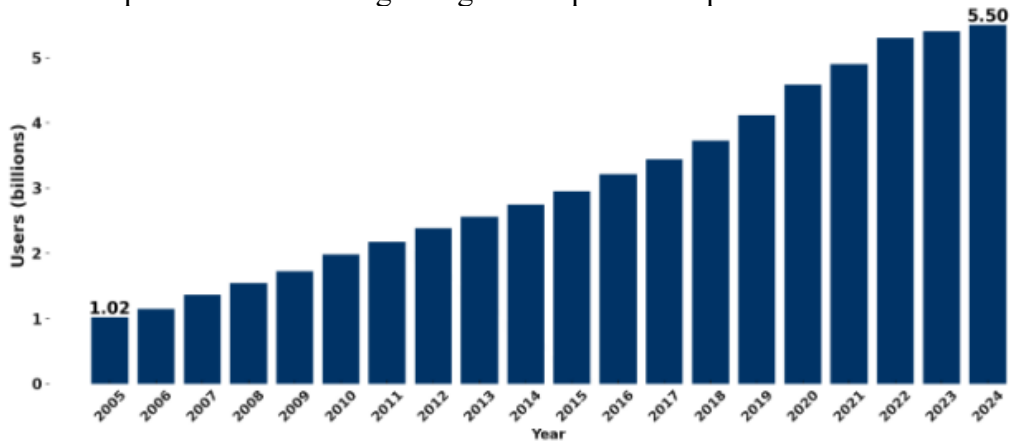


Figure 1. Number of Internet users worldwide, billions [1]

Under these conditions, visual and interactive solutions have become a key mechanism in the competition for users' attention. One of the first and most influential examples of this approach was The New York Times' report «Snow fall: the avalanche at tunnel creek» (2012), which utilized 3D animations, interviews, maps, and visualizations that were automatically activated as the reader scrolled down the page [2]. According to Source and OpenNews, the longread received over 3,5 million page views and around 2,9 million unique visitors during the first six days since being published, justifying the strong user interest in the multimedia format. The success of these projects demonstrated the potential of combining visual and interactive content in digital journalism, not merely enabling the transmission of information but also the building of an integrated audiovisual product that increases emotional engagement with the material.

Generally, the international media environment has experienced a deep change in the last two decades, passing from traditional media to digital media. Oberlo's data report that by 2025 the differential between time spent on traditional media and digital media in the USA is over three hours per day (fig. 2).

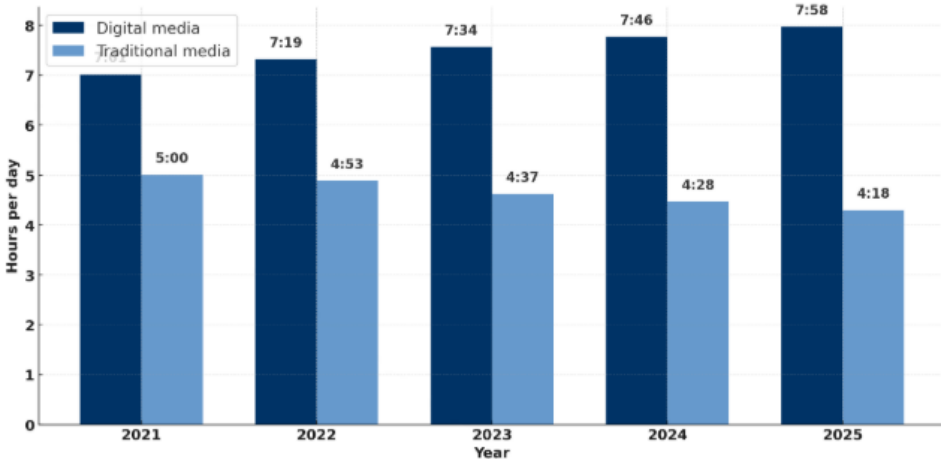


Figure 2. Comparison of time spent by users on media consumption [3]

This tendency clearly mirrors the general media market transformation from traditional content consumption to customized digital formats. Social media are not just leading communication platforms, but also leading news, entertainment, and advertising content providers that have successfully displaced print and television media in consumers' daily media consumption patterns.

At the institutional level, digitalization has contributed to a redefinition of professional roles within newsrooms [4]. Journalistic work is no longer solely text-centered: contemporary journalists engage with visual editor interfaces, make decisions regarding the structure and visual logic of content presentation, and participate in the development of interactive storytelling scenarios.

The automation of visual content production through the use of an embedded visual editor significantly streamlines editorial workflows. Key stages such as preparation, layout, and publication of multimedia materials can be carried out without developer involvement, thereby reducing the workload on technical teams and accelerating the overall production cycle (table 1).

Table 1

Comparative analysis of production workflow stages in traditional and automated models

Production stage	**Traditional approach (without visual editor)**	**Automated approach (with visual editor)**
Initiation and planning	Requires coordination between editors, designers, and the technical team to assess project feasibility.	Performed autonomously by the editorial team during internal prototyping using the editor interface.
Layout and composition	Executed manually by developers using HTML, CSS, and JavaScript.	Performed through a graphical interface without programming, using visual component configuration
Approval process	Involves multiple iterations between editorial, design, and development teams; delays are possible.	All changes are made and approved within a unified editorial system and workflow.
Testing and debugging	Requires testing of scripts, CMS compatibility, and cross-browser behavior.	Uses a unified rendering engine, which minimizes behavioral inconsistencies and reduces the need for testing.
Editing and revisions	Requires returning to the technical implementation stage and recompiling.	Editorial team retains access to the publication; changes can be made at any time via the editor interface.
Publication and integration	Requires manual embedding of code blocks into the CMS and	Publication is automated and integrated through API with the content management system.

	configuration of the execution environment.	

Thus, the shift from manual development to systems for visual content assembly reflects a broader trend in the evolution of digital media – from designing isolated publications to building a flexible and reusable production environment.

Architecture of the visual editor

The modern architecture of a visual editor for digital media is based on the principles of modularity, extensibility, and separation of concerns across application layers. A key design goal is to offer non-technical users – including editors, designers, and content creators – user-friendly tools for visual layout. The user interface adheres to **WYSIWYG** (What You See Is What You Get) concepts, tailored to media-specific processes: editable elements (e.g., cards, containers), animation triggers, timelines, and customizable visual settings are all set up through forms or direct interaction.

The **interface** is built upon a component-based framework that clearly differentiates functional logic from visual display. This improves extensibility by permitting the inclusion of additional widgets – such as custom blocks for data visualizations, 3D spaces, or multimedia players. Multimedia support employs standard browser APIs: Canvas is used for 2D graphics, WebGL for 3D content, and video elements are handled by HTML5 <video> tags along with wrappers for managing playback and synchronization. Interactive components – including maps, sliders, and polls – are encapsulated as self-contained modules with state management facilitated by a centralized data tree.

From an architectural perspective, the editor employs a **client-server model with clear roles.** The customer, a single-page application utilizing frameworks such as React or Vue, handles interactivity and local state management. The server manages storage, authentication, access rights, and version control. Content is saved in a serialized format (e.g., JSON), distinguishing structure from presentation and guaranteeing long-term adaptability.

A fundamental architectural element is a universal client-side script that displays content in the user's browser according to saved descriptions. The division of data, logic, and rendering facilitates simpler maintenance and scalability. Centralized updates, like adding new features or modifying rendering engines, necessitate no modifications to separate articles, boosting resilience and reducing ownership expenses.

Integration with digital media websites

Data exchange between the website and the visual editor is organized according to a distributed interaction model, utilizing RESTful API and webhook mechanisms. During the initialization phase, the website transmits the article ID, access parameters, current document structure and a callback URL to the editor. Upon saving, the editor generates a JSON document containing the full structure of the publication and notifies the website via the specified webhook, providing a URL from which all associated media assets can be downloaded. This architecture ensures a clear separation of responsibilities: the website remains the source of metadata, authorization and content distribution, while the editor functions as the interface for content editing (table 2).

Table 2

Formats and protocols of interaction between the editor and the website [5, 6]

Interaction stage	Protocol / method	Description
Session initialization	POST /api/session/open	The website sends the article ID, access parameters, and callback URL.
Resource loading	HTTPS (GET)	The editor fetches media assets from the site using deferred authorization.
Saving changes	Webhook (POST /callback)	The editor sends a JSON document with article data and a link to media resources.
Editor access	JWT / OAuth2	Authentication of editorial users via token-based authorization.
Content serialization	JSON or Markdown-like DSL	Structured article description for a universal rendering engine.

Visual publications, which may include scripts, external frames, and API interactions, need to be separated from the primary website environment. In practice, there are two main methods employed: incorporating content inside an iframe with limited settings (such as sandbox, CSP), and hosting interactive materials on a subdomain with a distinct execution context. CSP configurations are enforced by the visual editor for the published content, enabling strict control over script loading, inline code execution, and requests to external domains. This is especially important when working with user-generated content (UGC) systems, where the threat of introducing harmful code must be completely addressed (table 3).

Table 3

Execution environment isolation mechanisms

Isolation method	Technology	Application
Isolated iframe	sandbox, allow-scripts	Execution of interactive publications in a container without access to the website's DOM.
Subdomain with CORS	story.domain.com, Access-Control-Allow-Origin	Separation of security policies between the main site and the publication.
CSP	Content-Security-Policy	Restriction of inline scripts; control over external content sources.
Token-based access	JWT, HMAC, temporary tokens	Access control for publications and protection against cross-site request forgery (CSRF) attacks.

In the context of using visual stories on UGC platforms, it is essential to account for specific security threats associated with publishing content of unknown origin. Unlike articles created by in-house editorial teams, user submissions may contain potentially unsafe HTML, external scripts, or improperly formatted media files. To mitigate these risks, an automated moderation system is implemented, encompassing document structure analysis, HTML and CSS sanitization, link and media validation, and semantic text analysis.

Moderation can be performed either on the editor side (prior to submission) or on the website side (upon publication). When necessary, execution environments are sandboxed using containerization techniques with limitations on script execution time and memory usage, including the use of Web Workers and WebAssembly-based sandboxing. This approach ensures platform security while preserving the interactive capabilities of the visual editor.

Thus, the effective integration of a visual editor into a digital media infrastructure relies on strict adherence to architectural principles of distributed interaction, a multi-layered execution security model, and automated content moderation mechanisms.

Advertising effectiveness and monetization potential of interactive content

The development of visual editors within the digital media ecosystem contributes to the emergence of new monetization models based on a high degree of integration between advertising content and editorial formats. Unlike traditional formats – such as banners and static ad blocks – interactive publications enable the creation of full-scale branded projects in which commercial messages are seamlessly embedded into the logical and visual structure of the material, without disrupting the user experience.

According to Precedence Research, the global digital advertising market reached $600 billion in 2024. It is projected to exceed $1,483 billion by 2034, with an average annual growth rate of 9,47% during the 2025-2034 period (fig. 3).

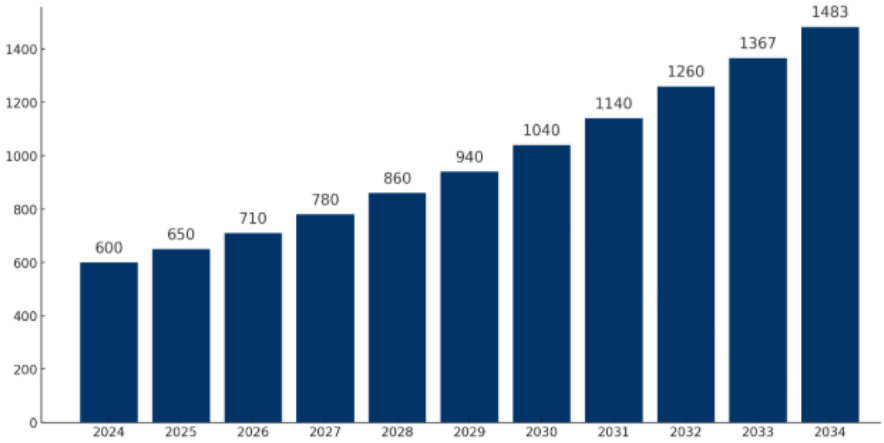


Figure 3. Global digital advertising market size, billion dollars [7]

This dynamic highlights a strategic shift in the advertising industry toward digital and personalized channels of engagement. Interactive formats are of particular interest in this context, as they deliver significantly higher levels of user engagement compared to traditional media. According to Mediafly, the average time spent interacting with interactive content is 13 minutes, compared to just 8,5 minutes for static content – demonstrating superior audience retention capabilities.

Interactive elements such as animated transitions, visual highlights, tabs, quizzes, and infographics enable the construction of personalized perception pathways in which branded content becomes an integrated part of the user experience [8]. The adaptive architecture enables consistent performance across various devices, it necessitates addressing the distinct requirements of desktop and mobile environments during the markup phase of interactive article development.

Performance measurements further confirm the advantages of interactive formats: the click-through rate (CTR) for native ads embedded in interactive publications reaches 0,2%, which is four times higher than the industry average for banner ads (0,05%). Moreover, 88% of marketers report that personalized interactive content helps them stand out from competitors and enhances the impact of advertising messages [9].

Current data on corporate intentions also affirm the high potential of interactive solutions. A 2025 survey conducted by Wynter among content marketing and SEO managers and directors found that 11,3% of respondents planned to invest more than $45,000 per month in content marketing – up from 4,1% in 2024. This indicates growing confidence in digital formats as cost-effective promotional channels.

Additional metrics further reflect increasing efficiency: according to a HubSpot survey of 336 SEO professionals and marketers in the U.S., the average click-through rate for SEO content is 13%, with a median of 8%, and users view an average of 7 pages per session. Furthermore, 43% of web analysts reported increased traffic to their primary websites in 2024 compared to the previous year, while only 14% noted a decline.

Taken together, these data points illustrate that interactive formats are no longer just a supplement to digital strategies, but have become a central component. By integrating visual, technical, and marketing advantages, they enable the development of a sustainable ecosystem of digital storytelling with high commercial returns. In today's oversaturated media environment – where user attention is the most valuable resource – interactivity and content personalization emerge as key drivers of communication effectiveness and audience loyalty.

Conclusion

Visual editors integrated into the infrastructure of digital media represent a technologically grounded tool capable of driving systemic transformation across editorial, user, and commercial aspects of media production. Through architectural modularity, support for universal interaction interfaces, and the separation of data presentation logic from storage layers, such systems reduce transactional costs, enhance the flexibility of the content cycle, and enable new monetization pathways. Amid the growing importance of visual and interactive content, the visual editor emerges

as a resilient element of the digital media ecosystem – combining engineering reliability with editorial efficiency.

References

1. Internet use continues to grow, but universality remains elusive, especially in low-income regions / ITU // URL: https://www.itu.int/itu-d/reports/statistics/2024/11/10/ff24-internet-use (date of access: 16.04.2025)
2. McHugh S. The narrative podcast as digital literary journalism: Conceptualizing S-Town // Literary Journalism Studies. 2021. Vol. 13. No. 1-2. P. 100-29.
3. US Media Consumption (2021-2025) / Oberlo // URL: https://www.oberlo.com/statistics/us-media-consumption (date of access: 18.04.2025)
4. Amzin A., Galustyan A., Gatov V., Castells M., Kulchitskaya D., Loseva N., Parks M., Paranko S., Silantyeva O., van der Haak B. (2016). How new media changed journalism: 2012-2016. Edited by S. Balmayeva & M. Lukina. Yekaterinburg: Humanitarian University. 304 p.
5. Drogunova Y. Integration of UI and API testing into CI/CD processes as a factor in accelerating the release of digital products // Universum: technical sciences: electron. scientific journal. 2025. No. 5(134). P. 26-29.
6. Terletska K. Architecting event-driven stream processing: Technologies for reliability and analytical availability in distributed computing systems // International Journal of Scientific Research and Engineering Development. 2025. Vol. 8(3). P. 667-671
7. Digital Ad Spending Market Size and Forecast 2025 to 2034 / Precedence Research // URL: https://www.precedenceresearch.com/digital-ad-spending-market (date of access: 19.04.2025)
8. Makhtibekov A. Effectiveness of multichannel marketing in the context of digital transformation // Universum: economics and law: electronic scientific journal. 2025. No. 4(126). P. 15-19.
9. Interactive Ads in 2024: Engaging Audiences Creatively / Bannerflow // URL: https://www.bannerflow.com/blog/interactive-ads-engaging-audiences (date of access: 19.04.2025).

UDC 004.415

MICROSERVICE ARCHITECTURES FOR FINANCIAL PLATFORMS: CHALLENGES AND SOLUTIONS

Nasyrova I.N.
bachelor's degree, University of Helsinki (Helsinki, Finland)

АРХИТЕКТУРЫ МИКРОСЕРВИСОВ ДЛЯ ФИНАНСОВЫХ ПЛАТФОРМ: ВЫЗОВЫ И РЕШЕНИЯ

Насырова И.Н.
бакалавр, Хельсинкский университет (Хельсинки, Финляндия)

Abstract

This paper explores the architectural and operational complexities of implementing microservice-based architectures (MSAs) in financial platforms. It investigates key challenges related to modular service decomposition, inter-service communication, data consistency, and security enforcement, with particular focus on high-assurance environments. Emphasis is placed on hybrid design patterns, including event-driven coordination, fault isolation, and observability-driven scaling, which enable resilience and regulatory compliance. The analysis is supported by diagrams and tabular comparisons illustrating practical configurations. The findings aim to guide the development of scalable, auditable, and fault-tolerant financial systems capable of sustaining real-time operations in dynamic conditions.

Keywords: microservices, financial platforms, distributed systems, event-driven architecture, data consistency, observability, fault tolerance, security.

Аннотация

В статье рассматриваются архитектурные и эксплуатационные особенности внедрения микросервисных архитектур (MSAs) в финансовые платформы. Проанализированы ключевые проблемы, связанные с модульной декомпозицией, межсервисной коммуникацией, обеспечением согласованности данных и реализацией распределённых механизмов безопасности в условиях высоких регуляторных требований. Особое внимание уделено гибридным подходам, включающим событийное взаимодействие, изоляцию отказов и масштабирование на основе наблюдаемости. Представлены диаграммы и сравнительные таблицы, иллюстрирующие практические конфигурации. Полученные результаты ориентированы на разработку масштабируемых, отказоустойчивых и проверяемых финансовых систем, адаптированных к динамичным условиям эксплуатации.

Ключевые слова: микросервисы, финансовые платформы, распределённые системы, событийная архитектура, согласованность данных, наблюдаемость, отказоустойчивость, безопасность.

Introduction

The growing complexity of financial systems, coupled with demands for agility, resilience, and regulatory compliance, has driven a shift from traditional monolithic architectures to modular, microservice-based designs. Financial platforms today operate under conditions of high transaction throughput, stringent latency requirements, and constant integration with heterogeneous external systems, including payment gateways, identity providers, and regulatory databases. Monolithic systems, while historically dominant, struggle to scale horizontally, adapt to evolving business logic, or isolate failures effectively, thus posing a significant risk in dynamic financial ecosystems.

Microservice architectures (MSAs) offer a compelling alternative by decomposing large applications into independent, loosely coupled services that can be developed, deployed, and scaled independently. This paradigm enables financial institutions to implement domain-driven design (DDD), embrace DevOps practices, and respond swiftly to changes in compliance or market behavior. However, the transition to MSAs introduces architectural complexity, increased operational overhead, and non-trivial challenges in service orchestration, data consistency, and security enforcement. For financial applications, these challenges are exacerbated by high sensitivity to downtime, transaction integrity, and real-time observability.

This paper aims to systematically analyze the architectural and operational challenges associated with adopting microservice architectures in financial platforms. Key focus areas include modular service decomposition, fault tolerance, inter-service communication patterns, data synchronization strategies, and security models. In addition to highlighting typical bottlenecks and failure domains, the paper presents visual models and tabular evaluations of microservice performance characteristics under financial constraints. The findings are intended to inform the design of resilient, auditable, and compliant microservice ecosystems tailored for high-assurance financial environments.

Main part

Modular service decomposition and domain alignment

In microservice architectures, the effectiveness of system modularization directly influences scalability, resilience, and maintainability. For financial platforms-characterized by complex business domains and high regulatory oversight-modular decomposition must reflect clear domain boundaries to ensure traceability, autonomy, and auditability of each component.

Domain-driven design provides a theoretical and practical foundation for achieving this alignment. By organizing services around bounded contexts, development teams can encapsulate business logic and data within well-defined modules, such as Account Management, Fraud Detection, Payment Processing, or Regulatory Reporting. Each module can evolve independently, simplifying compliance updates and reducing the blast radius of failures. Figure 1 illustrates a sample domain decomposition for a retail banking platform, highlighting how services are structured according to core business functions.

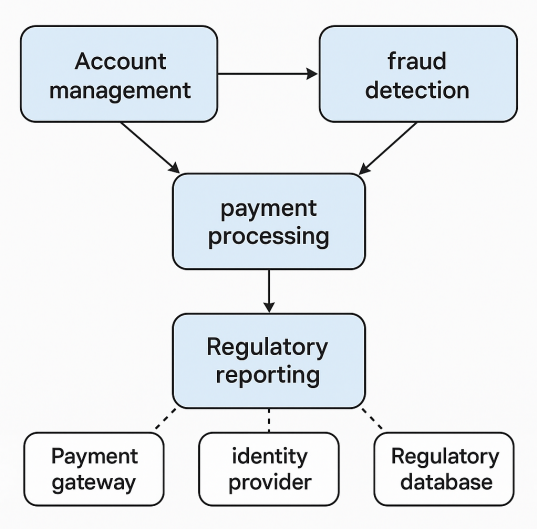


Figure 1. Modular decomposition of a financial platform based on domain-driven design

Additionally, the principle of single responsibility within each microservice mitigates codebase sprawl and facilitates the use of targeted, domain-specific technologies. For example, services handling high-throughput payment requests may be written in low-latency languages (e.g., Go or Rust), while reporting modules can leverage data-oriented platforms (e.g., Apache Spark or ClickHouse) [1]. However, an over-fragmented decomposition can lead to excessive inter-service communication and increase the cognitive load on developers and operators. Therefore, financial institutions must strike a balance between granularity and cohesion.

Another critical factor is the consistent mapping of business capabilities to service contracts and APIs. In regulated environments, each exposed endpoint must adhere to strict data handling policies and provide deterministic behavior under load. Failure to standardize these interfaces not only introduces integration risks but may violate compliance requirements, especially under data protection and financial audit regulations.

Inter-service communication: patterns, trade-offs, and fault isolation

In MSAs, the method by which services communicate with each other is a critical design decision that directly impacts system latency, reliability, and maintainability. For financial platforms-where even milliseconds of delay or transaction failures can lead to regulatory violations or monetary loss-communication patterns must be carefully selected, implemented, and monitored [2].

Two primary modes of inter-service communication exist: synchronous (typically HTTP/gRPC APIs) and asynchronous (via message brokers such as Apache Kafka, RabbitMQ, or NATS). Synchronous communication provides simplicity and immediacy but introduces tight temporal coupling. A failure in a downstream service can cascade and block upstream requests, degrading system availability. Asynchronous communication, on the other hand, enables better fault tolerance and elasticity, decoupling service lifecycles and smoothing traffic bursts. However, it increases system complexity and requires robust event tracking, message deduplication, and retry policies.

For financial systems, a hybrid approach is often employed. Time-sensitive user interactions-such as account balance queries or KYC verification-are executed synchronously, while transactional workflows like payment orchestration or anti-fraud checks are designed using asynchronous patterns with event sourcing and eventual consistency guarantees. This dual-mode strategy ensures both responsiveness and resilience.

Figure 2 illustrates a hybrid communication architecture in a financial microservice environment. Core business services interact via RESTful APIs for real-time operations, while transactional and analytical components communicate through a distributed event bus. Failover queues, idempotent endpoints, and retry logic are incorporated to prevent message loss or duplication in critical processes.

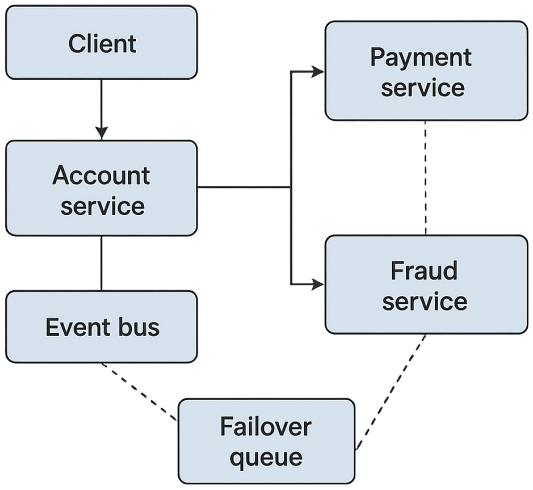


Figure 2. Hybrid communication architecture in a microservice-based financial platform

The choice of communication mechanism also affects observability. Distributed tracing systems (e.g., OpenTelemetry, Jaeger) must be implemented to trace transactions across service boundaries and identify latency hotspots or failure points. In the context of financial auditing and compliance, detailed trace logs become essential components of forensic analysis.

Moreover, integrating observability tools into the communication infrastructure allows for proactive anomaly detection and adaptive service scaling [3]. Metrics such as request latency, error rate, queue depth, and circuit breaker activations can be aggregated using platforms like Prometheus or Datadog. These indicators help operations teams respond to degradations before they escalate into full-scale outages, which is particularly crucial in high-assurance financial environments.

To ensure end-to-end traceability, correlation identifiers (e.g., trace IDs, span IDs) must propagate across all synchronous and asynchronous communication channels. Without consistent metadata propagation, it becomes difficult to reconstruct distributed transaction chains-a significant limitation in post-incident reviews or compliance audits.

Finally, communication resilience must be validated through chaos engineering practices. Simulated failures such as delayed messages, dropped connections, or misrouted events help uncover hidden dependencies and test fallback mechanisms in real conditions. Such proactive validation is indispensable for achieving high availability targets (e.g., 99.99%) in financial ecosystems where downtime equates to revenue loss and reputational damage.

Data synchronization and consistency through event-driven architecture

Ensuring consistency and synchronization across distributed services is one of the central challenges in microservice-based financial platforms. Unlike monolithic systems, where data integrity can be maintained through tightly coupled ACID transactions, microservice architectures operate in an environment where each service manages its own data store and evolves independently. This architectural decoupling introduces the risk of data divergence, which can be particularly damaging in financial applications [4].

To mitigate this risk, financial systems increasingly rely on event-driven communication as the foundation for achieving eventual consistency. Rather than invoking services directly in a tightly synchronous chain, each service reacts to events published on a shared event bus, allowing for loose coupling and asynchronous propagation of state changes. This approach decouples the execution flow and eliminates blocking dependencies, improving system resilience and scalability.

In the context of financial transactions, services such as customer management, order processing, payment authorization, and notification delivery operate independently but remain logically coordinated through events. For example, an orders service may emit an OrderPlaced event, which triggers downstream actions by the payments and notifications services. These services, in turn, emit events like PaymentProcessed or NotificationSent, enabling other components to react accordingly. This model supports auditability and observability while avoiding the complexity and fragility of distributed locking or two-phase commits.

Figure 3 illustrates this architecture, where core domain services interact exclusively via an event bus to exchange business-relevant events. Each module consumes only the events it subscribes to, ensuring logical separation of concerns, operational independence, and traceable state transitions. The architecture supports high-frequency financial workflows while maintaining consistency guarantees under load.

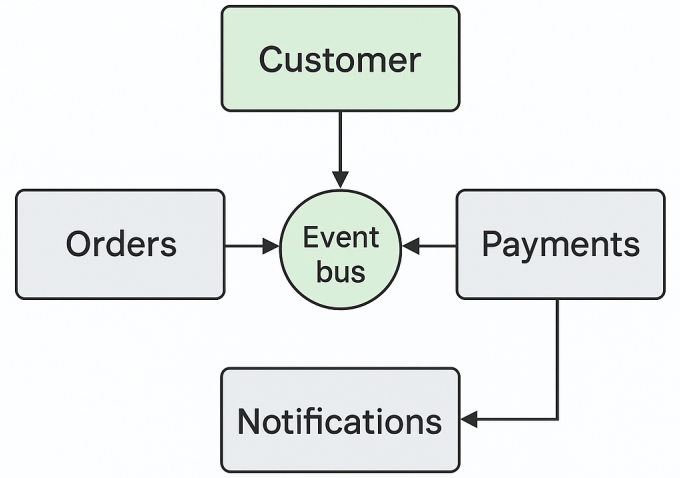


Figure 3. Data synchronization in a microservice-based financial system using an event-driven approach

The architecture presented in figure demonstrates how event-driven communication enables scalable and resilient data synchronization in microservice-based financial platforms. By decoupling services and coordinating actions through a shared event bus, the system achieves eventual consistency without relying on centralized transaction management. This design not only enhances

fault tolerance and throughput under high load but also facilitates modular auditing, recoverability, and compliance tracking-core requirements in regulated financial environments.

Security enforcement in microservice-based financial environments

Security is a critical concern in financial microservice architectures, where data flows across numerous independently deployed services, often spanning cloud-based and on-premises infrastructures. Unlike monolithic systems, where centralized security policies can be enforced more easily, microservice environments demand distributed security mechanisms that are consistent, scalable, and compliant with strict regulatory standards such as PCI DSS, PSD2, and GDPR [5].

A core principle in secure microservice design is the zero trust model, which assumes that no service-internal or external-should be inherently trusted. All communication between services must be authenticated, authorized, and encrypted, regardless of whether it occurs within the same data center or across cloud boundaries. This is typically implemented through mutual TLS (mTLS) for service-to-service authentication, combined with token-based authorization protocols like OAuth 2.0 and JSON Web Tokens (JWTs) [6].

Fine-grained access control is another critical component. Role-based access control (RBAC) and attribute-based access control (ABAC) must be enforced at the service level to ensure that each operation is only accessible to authorized users or services. Policy engines such as OPA (Open Policy Agent) can be integrated to manage these rules declaratively and consistently across the architecture.

Furthermore, secrets management is essential to prevent credential leakage and unauthorized access. Services must not embed credentials in source code or configuration files. Instead, secure vaults (e.g., HashiCorp Vault, AWS Secrets Manager) should be used to manage dynamic secrets with short lifespans and granular access scopes.

Monitoring and auditing mechanisms must also be embedded across the system. Every security-relevant event-such as failed authentications, permission denials, or unusual request patterns-should be logged and correlated through a centralized security information and event management (SIEM) system. In regulated financial contexts, audit trails must not only be comprehensive but also tamper-resistant and readily exportable for compliance review.

Finally, threat modeling and vulnerability scanning should be incorporated into the DevSecOps lifecycle. Static and dynamic analysis tools (SAST/DAST), container image scanning, and dependency checks ensure that vulnerabilities are caught before they reach production [7]. Financial platforms must implement security as code, continuously validating the system's resilience against evolving threats and attack vectors.

Scalability and fault tolerance strategies in financial microservice platforms

In the context of financial operations, high availability and elastic scalability are not merely desirable attributes-they are essential for maintaining service continuity, regulatory compliance, and customer trust. MSAs inherently support these qualities through modular deployment and independent scaling. However, realizing effective fault tolerance and scalability in production requires a deliberate combination of architectural patterns, infrastructure tooling, and runtime policies [8].

Horizontal scaling is a primary advantage of MSAs, allowing individual services to scale out based on demand without affecting the rest of the system. Services responsible for high-frequency operations-such as payment gateways, real-time fraud detection, or account lookups-can be deployed across multiple replicas and managed by orchestrators like Kubernetes, which dynamically allocates resources in response to system metrics.

To ensure fault isolation, services are typically deployed in separate containers or pods, preventing failures in one component from cascading into others. Circuit breakers, timeouts, and retries are implemented to detect and contain faults locally, while service meshes (e.g., Istio, Linkerd) provide observability and control over traffic flow and failure recovery [9].

A complementary mechanism is graceful degradation, which ensures that non-critical features can fail without compromising core functionality. For instance, if the notifications service fails, payment confirmation can still proceed, deferring message delivery for later processing. Such design decisions are vital for user experience and operational continuity during partial outages.

The table 1 below summarizes key strategies for scalability and fault tolerance, along with their respective benefits and trade-offs in financial systems.

Table 1

Scalability and fault tolerance strategies in microservice-based financial platforms

Strategy	Purpose	Implementation tools	Trade-offs
Horizontal scaling	Increase throughput	Kubernetes, Docker Swarm	Resource cost, coordination complexity
Circuit breakers	Prevent cascading failures	Hystrix, Resilience4j	Requires tuning to avoid false positives
Retry with backoff	Handle transient faults	Spring Retry, Polly	May delay recovery in case of real failures
Graceful degradation	Preserve core functionality	Custom logic, fallback responses	Degraded UX, complexity in failure mapping
Service mesh	Control traffic and recovery	Istio, Linkerd	Added latency, increased configuration burden
Auto-scaling policies	Elastic resource management	HPA, KEDA, AWS Auto Scaling	Dependent on accurate metrics

These strategies must be tuned to the specific needs of financial workflows, where real-time response, regulatory auditability, and transactional accuracy cannot be compromised. Hybrid approaches that blend infrastructure-level automation with domain-specific fallback logic offer the most reliable path toward resilient and scalable systems.

While theoretical frameworks provide a foundation, real-world financial systems often rely on hybrid resilience patterns that combine multiple mechanisms tailored to the business context. For instance, a high-frequency trading platform may prioritize low-latency communication and local state caching to maximize speed, while a digital bank may emphasize transactional durability and multi-region failover to meet service-level agreements (SLAs).

In such systems, chaos engineering has become a vital practice for validating fault tolerance under production-like conditions. By intentionally injecting failure scenarios-such as service unavailability, network partitions, or delayed dependencies-teams can evaluate the effectiveness of their fallback logic and alerting systems. This approach not only reveals architectural weaknesses but also trains operational teams for incident response in high-stakes environments [10].

Another essential factor is observability-driven scaling. Unlike naive resource-based scaling (e.g., CPU/memory usage), financial platforms benefit from behavioral indicators such as transaction volume, fraud alert frequency, or latency spikes in specific flows. Integrating these business-level signals into autoscaling triggers enables more intelligent and context-aware resource management, reducing both cost and risk.

Finally, resilient financial architectures increasingly incorporate multi-zone and multi-cloud deployments to avoid single points of failure. By distributing critical services across isolated failure domains, systems can recover quickly from infrastructure outages or cloud-specific disruptions. However, these setups require careful coordination of data replication, consistent configuration management, and latency-aware routing.

Together, these practices enable financial microservice platforms not only to withstand disruption but also to adapt dynamically under load-supporting real-time processing, continuous uptime, and regulatory accountability in volatile operational environments.

Conclusion

The transition from monolithic systems to microservice architectures represents a paradigm shift in the design of financial platforms, driven by the need for agility, scalability, and regulatory alignment. While MSAs offer substantial advantages-including modular deployment, autonomous

scaling, and enhanced fault isolation-they also introduce non-trivial challenges in service coordination, data consistency, security enforcement, and operational resilience.

This paper has examined the critical architectural and operational considerations for implementing microservices in financial systems. It has highlighted how domain-aligned decomposition, hybrid communication models, event-driven consistency mechanisms, and distributed security controls collectively form the backbone of resilient financial microservice ecosystems. Additionally, the analysis underscored the importance of observability, failover strategies, and real-world resilience practices such as chaos engineering and multi-cloud redundancy.

Successful adoption of MSAs in the financial sector requires more than technical reengineering; it demands a holistic approach that aligns infrastructure design, business logic segmentation, compliance needs, and runtime governance. By embracing hybrid strategies that balance performance with auditability, and automation with domain sensitivity, financial institutions can build platforms that not only scale under pressure but also maintain integrity, availability, and trust in increasingly complex operational landscapes.

References

1. Söylemez M., Tekinerdogan B., Kolukısa Tarhan A. Challenges and solution directions of microservice architectures: A systematic literature review // Applied sciences. 2022. Vol. 12. No. 11. P. 5507.
2. Bhatnagar S. Cost optimization strategies in fintech using microservices and serverless architectures // Computing. 2025. Vol. 19. No. 01.
3. Muley Y. Comparative Analysis of Monolithic and Microservices Architectures in Financial Software Development // J Artif Intell Mach Learn & Data Sci 2024. 2024. Vol. 2. No. 4. P. 1846-1848.
4. Kovalenko A. Architectural and algorithmic methods for enhancing the resilience of high-load backend services in the financial sector // Norwegian Journal of development of the International Science. 2025. №158. P. 87-91.
5. Mobit I.A.C. Technological, organizational, and environmental factors and the adoption of microservices in the financial services sector. Robert Morris University. 2023.
6. Yan W., Shuai F. Application of microservice architecture in commodity erp financial system // International Journal of Computer Theory and Engineering. 2022. Vol. 14. No. 4.
7. Malali N. Microservices in life insurance: enhancing scalability and agility in legacy systems // International Journal of Engineering Technology Research & Management (IJETRM). 2022. Vol. 6. No. 03.
8. Driss M., Hasan D., Boulila W., Ahmad J. Microservices in IoT security: current solutions, research challenges, and future directions // Procedia Computer Science. 2021. Vol. 192. P. 2385-2395.
9. Oumoussa I., Faieq S., Saidi R. When Microservices Architecture and Blockchain Technology Meet: Challenges and Design Concepts // International Conference on Advanced Technologies for Humanity. Cham: Springer International Publishing. 2021. P. 161-172.
10. Siddiqui H., Khendek F., Toeroe M. Microservices based architectures for IoT systems-state-of-the-art review // Internet of Things. 2023. Vol. 23. P. 100854.

UDC 004.42

USE OF GRAPH DATABASES FOR USER BEHAVIOR ANALYSIS

Bastrykin Y.L.
master's degree, University of Edinburgh
(Edinburgh, United Kingdom)

Yumasheva N.B.
master's degree, University of Edinburgh
(Edinburgh, United Kingdom)

ИСПОЛЬЗОВАНИЕ ГРАФОВЫХ БАЗ ДАННЫХ ДЛЯ АНАЛИЗА ПОЛЬЗОВАТЕЛЬСКОГО ПОВЕДЕНИЯ

Бастрыкин Ю.Л.
магистр, Эдинбургский университет
(Эдинбург, Великобритания)

Юмашева Н.Б.
магистр, Эдинбургский университет
(Эдинбург, Великобритания)

Abstract

This paper investigates the application of graph databases for user behavior analysis, highlighting their advantages over relational models in representing and querying complex interaction patterns. Key aspects explored include data modeling techniques, query strategies using Cypher, integration into analytics pipelines, graph algorithm use cases, and system performance comparisons. Visual analyses and empirical benchmarks demonstrate the efficiency of graph-native operations in behavioral contexts, particularly for multi-hop queries and influence modeling. The study also addresses operational challenges and outlines emerging trends such as graph machine learning and temporal graph modeling. The results support the adoption of graph databases as a core component of intelligent, relationship-aware analytical systems.

Keywords: Graph databases, user behavior analysis, Cypher queries, graph algorithms, data modeling, performance comparison, behavior analytics, temporal graphs.

Аннотация

В статье рассматривается применение графовых баз данных для анализа поведения пользователей и подчёркиваются их преимущества перед реляционными моделями при работе с многосвязными структурами взаимодействий. Проанализированы подходы к моделированию данных, реализация запросов на языке Cypher, интеграция в аналитические пайплайны, применение графовых алгоритмов и сравнительная оценка производительности систем. На основе визуализаций и эмпирических тестов показана эффективность графового подхода в поведенческих сценариях, особенно при работе с глубокой связностью и моделированием влияния. Также обсуждаются эксплуатационные ограничения и ключевые перспективы, включая графовое машинное обучение и временные графы. Полученные результаты подтверждают целесообразность внедрения графовых СУБД как основы для построения интеллектуальных и ориентированных на отношения аналитических платформ.

Ключевые слова: Графовые базы данных, анализ поведения пользователей, Cypher-запросы, графовые алгоритмы, моделирование данных, сравнение производительности, поведенческая аналитика, временные графы.

Introduction

In recent years, the exponential growth of digital interaction data has driven the demand for more sophisticated tools to capture, represent, and analyze user behavior. Traditional relational databases, while effective for structured and tabular data, often struggle to model the complex, interconnected patterns inherent in user activity logs, social media footprints, and recommendation systems. This limitation has led to an increased interest in graph-based data representations, where entities and their relationships are treated as first-class citizens in the data model.

Graph databases (GDBs), such as Neo4j, Amazon Neptune, and ArangoDB, offer native support for relationship-centric data structures, enabling efficient traversal and pattern matching. Unlike relational models that rely heavily on joins, GDBs provide direct edge-based connections between nodes, significantly reducing query latency when exploring behavioral paths or networked interactions. These features make them especially suitable for applications involving user segmentation, fraud detection, influence mapping, and personalization engines, where the semantics of relationships are as critical as the attributes of individual users.

This paper aims to examine the practical application of graph databases in user behavior analysis, focusing on modeling techniques, query strategies, and performance trade-offs. The study compares graph-based and relational approaches using real-world datasets and demonstrates how graph algorithms-such as community detection, centrality metrics, and path analysis-can extract meaningful behavioral patterns. Through visual models, query examples, and benchmarking tables, the paper outlines best practices for designing graph-powered analytical workflows in dynamic digital environments.

Main part. **Modeling user behavior data in relational and graph databases**

Modeling user behavior requires not only capturing discrete user actions, but also representing the relationships and dependencies between those actions. In relational database systems, this typically involves multiple normalized tables and foreign key constraints, where joins are used to reconstruct interaction histories. However, as the complexity and density of relationships increase-such as in social graphs, clickstreams, or recommendation engines-the performance and clarity of relational models degrade rapidly.

Graph databases offer an alternative paradigm in which users, actions, sessions, and resources are represented as nodes, and the relationships between them (e.g., «clicked», «viewed», «follows», «purchased») are stored as edges. This structure allows for more intuitive modeling and enables recursive traversal operations with significantly lower computational cost. Queries that would require nested joins in SQL can often be expressed as short, expressive traversal patterns in graph query languages such as Cypher or Gremlin [1].

The following table 1 summarizes the key conceptual and operational differences between relational and graph-based approaches to user behavior modeling.

Table 1

Comparison of relational and graph database models for user behavior analysis

Aspect	Relational model	Graph model
Data structure	Tables with rows and foreign keys	Nodes and edges representing entities and relationships
Relationship representation	Indirect (via joins)	Direct (via edges)
Query complexity	High for multi-hop relationships	Low for recursive traversal
Performance with deep links	Degrades with number of joins	Stable with graph traversal

Aspect	Relational model	Graph model
Schema flexibility	Rigid, predefined schemas	Schema-optional, supports heterogeneous data
Use cases	Transactional systems, structured tabular data	Behavioral analysis, recommendations, social networks

This comparison illustrates that graph databases provide significant advantages in scenarios where relationship depth and query flexibility are critical. As user behavior increasingly manifests in multi-layered and temporal patterns, adopting graph-based models becomes essential for building accurate, real-time analytics pipelines.

Graph query strategies for user behavior analysis

Graph databases offer powerful querying mechanisms that go beyond traditional filtering and aggregation. User behavior analysis often requires tracing interactions across multiple degrees of separation-such as identifying chains of content consumption, influence paths in social networks, or anomalous navigation patterns. Graph query languages like Cypher (used in Neo4j) and Gremlin (used in TinkerPop-based systems) provide native support for recursive traversals, subgraph pattern matching, and graph algorithms-all of which are essential for behavioral insights.

One of the most common techniques is path traversal, used to identify the sequence of actions taken by a user or to discover how different users are connected through shared interactions [2]. For example, detecting a community of users who consistently follow similar purchase paths or identifying influence chains in referral-based ecosystems. Graph databases can execute such queries in linear time relative to path depth, while in relational systems this often results in multiple nested joins and exponential growth in execution time.

Another frequent approach is the use of centrality metrics (e.g., PageRank, betweenness, closeness) to identify the most influential users or sessions within a network. In behavioral contexts, high-centrality nodes may represent key navigational hubs, referral generators, or fraudulent actors. Similarly, community detection algorithms (e.g., Louvain, label propagation) help cluster users into behaviorally similar groups for segmentation or targeting.

To highlight the performance advantage of graph databases for multi-hop queries, the figure below compares the average query response times in a synthetic dataset as the relationship depth increases from 1 to 6 hops. As shown in figure 1, while graph-based traversal scales linearly, relational joins exhibit exponential degradation.

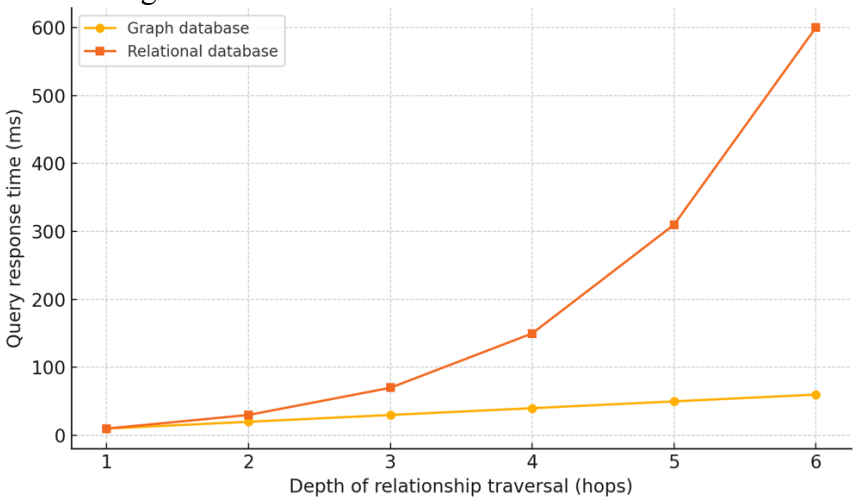


Figure 1. Query response time by depth of relationship traversal in graph vs. relational database models

The results illustrated in Figure clearly demonstrate the scalability advantage of graph databases for multi-hop relationship queries. As the traversal depth increases, the response time in relational systems grows exponentially due to the compounding cost of join operations. In contrast, graph databases maintain near-linear performance, enabling efficient exploration of deeply connected user

behavior patterns. This characteristic makes graph-based models particularly suitable for real-time behavioral analytics, where low-latency insights across complex interaction chains are required.

Graph algorithms for extracting behavioral patterns

Graph databases support a wide range of algorithms that can reveal latent behavioral structures not easily accessible through traditional data analysis techniques. These algorithms enable analysts to uncover user clusters, detect anomalies, evaluate influence, and optimize recommendation strategies based on structural properties of the user interaction graph [3].

One widely used category is community detection algorithms, such as the Louvain method or label propagation. These help to identify groups of users who exhibit similar behavioral trajectories-visiting similar sequences of pages, reacting to the same content, or purchasing related products. Such clustering is valuable for audience segmentation, personalization, and targeted marketing.

Another critical class is centrality algorithms, including PageRank, degree centrality, betweenness, and closeness. These metrics quantify the importance of nodes within the network. In behavioral contexts, central nodes may correspond to super-users, content hubs, or actors involved in suspicious activity propagation.

Similarity and proximity algorithms, such as Jaccard similarity or personalized PageRank, can identify users with shared interests or patterns, supporting collaborative filtering and social recommendations. For anomaly detection, graph outlier detection methods identify structurally rare patterns, such as unexpected edge density or users disconnected from typical interaction flows.

The table 2 below outlines common categories of graph algorithms, their primary goals, and examples of behavioral analysis use cases.

Table 2

Graph algorithms for user behavior analysis

Algorithm type	Primary goal	Example use cases
Community detection	Cluster users with similar behavior	Segmenting audiences, recommending group-based content
Centrality metrics	Identify influential nodes	Detecting super-users, fraud hubs, or key referrers
Similarity/proximity	Find structurally similar users	Collaborative filtering, social recommendations
Pathfinding/traversal	Discover behavioral chains	Navigation flow analysis, content funnel optimization
Outlier detection	Identify anomalous user patterns	Fraud detection, bot activity identification

As summarized in table, graph algorithms provide a versatile analytical toolkit for modeling and interpreting complex user behavior patterns. Each category of algorithms serves a distinct purpose-ranging from identifying communities and influential users to detecting anomalies and reconstructing behavioral paths. The ability to apply these algorithms directly within graph databases enables real-time, relationship-aware analytics that traditional systems often struggle to achieve. By selecting appropriate algorithms aligned with specific behavioral objectives, analysts can uncover hidden structures, personalize user experiences, and improve decision-making in dynamic digital environments.

Implementing behavior analysis using Cypher queries

Cypher, the declarative query language for Neo4j and other property graph databases, enables analysts to describe complex patterns of user behavior in a concise and expressive way. Rather than relying on cumbersome SQL joins, Cypher operates natively on node–relationship structures, making it particularly effective for analyzing sequences, cycles, and multi-hop interactions.

In behavioral analytics, Cypher is often used to implement three major classes of queries: path-based queries, structural ranking, and pattern discovery. Path-based queries are frequently employed

to trace navigation flows, user conversion funnels, or repeated access loops [4]. Structural ranking is useful for identifying high-impact users, such as referrers or hubs in content networks, while pattern discovery is used for detecting suspicious behavior, churn indicators, or anomalous interaction graphs.

The flexibility of Cypher allows analysts to combine conditions on both node attributes (e.g., user age, session duration) and relationship properties (e.g., frequency, time intervals), enabling fine-grained filtering of behavior. This makes it a powerful tool for segmenting user groups based on interaction types, recency, or cross-platform activity.

To better understand the practical use of Cypher in behavior analysis, figure 2 presents the relative frequency of different query types across a sample of real-world graph-based analytics workloads. The data illustrates that path queries dominate most use cases, followed by influence/ranking queries, with anomaly detection and temporal pattern matching growing steadily in adoption.

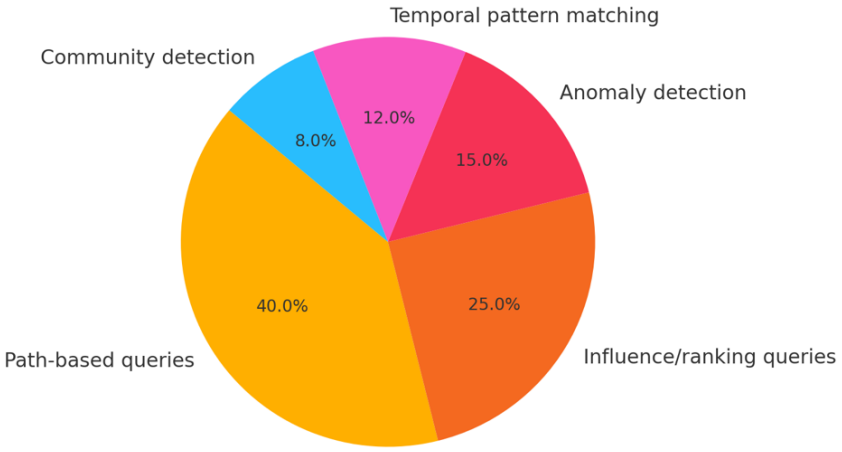


Figure 2. Distribution of Cypher query types in user behavior analysis use cases

As shown in figure, path-based queries constitute the largest share of Cypher-based behavior analysis workloads, reflecting their central role in tracing user navigation, conversion paths, and repeated interaction sequences. Influence and ranking queries are also widely adopted, particularly in use cases involving social graphs, referral systems, and recommendation optimization. The growing use of anomaly detection and temporal pattern matching indicates a shift toward more predictive and adaptive analytical approaches. This distribution highlights the flexibility of Cypher in supporting both descriptive and advanced behavioral modeling tasks within graph databases.

Integrating graph databases into behavior analytics architectures

While GDBs provide a powerful foundation for modeling and analyzing complex user interactions, their effective use in production requires seamless integration into broader data processing and analytics pipelines. In contemporary data-driven environments, behavioral data is typically generated across multiple systems-web logs, mobile applications, customer relationship management (CRM) platforms-and must be aggregated, normalized, and enriched before meaningful analysis can be performed. Integrating GDBs into this landscape demands careful orchestration of data flows, performance tuning, and alignment with organizational data governance policies.

One of the most common integration patterns involves extract–transform–load (ETL) pipelines that move behavioral event data from transactional sources into the graph store. Tools such as Apache NiFi, Kafka Connect, and Neo4j's own Data Importer enable automated ingestion of session logs, interaction events, and user profiles. Preprocessing steps may include timestamp normalization, deduplication, and enrichment with metadata (e.g., device type, location, user segment). A consistent data model must be designed to accommodate both entity diversity and evolving relationship schemas [5].

Once ingested, graph databases often operate alongside other analytical systems. Hybrid architectures may combine GDBs for relationship modeling with columnar databases (e.g., ClickHouse, BigQuery) for high-speed aggregations or with document stores (e.g., MongoDB) for

flexible session metadata storage. Business intelligence (BI) platforms can connect to GDBs via Cypher or GraphQL interfaces, while machine learning (ML) workflows increasingly incorporate graph embeddings and topological features derived from GDBs to improve model accuracy [6].

Despite these advantages, integration also poses challenges. Maintaining data consistency across systems, ensuring low-latency synchronization, and managing schema evolution in dynamic environments require ongoing architectural and operational effort. Additionally, access control, auditing, and compliance constraints must be enforced across all interconnected components, especially in industries like finance and healthcare.

Nonetheless, when properly integrated, graph databases significantly enhance the depth and contextual quality of behavioral analytics, enabling systems that go beyond static segmentation toward truly relational, adaptive, and intelligent user modeling.

Performance comparison of graph and relational databases in behavior analytics

Selecting the appropriate database architecture for behavior analytics is not merely a matter of data modeling preferences-it directly impacts system scalability, query latency, and analytical flexibility [7]. To evaluate this, a series of benchmark tests were conducted using representative behavior analysis tasks, including multi-hop traversal, user influence detection, and session pattern extraction. Both relational and graph database platforms were tested on equivalent datasets with controlled dimensions.

The results consistently indicate that graph databases outperform relational systems in queries involving deep relationship chains and structural computations. In contrast, relational databases maintain an advantage in flat aggregations and predefined tabular reporting. The performance gap widens with query complexity, especially as the number of joins in SQL increases beyond three or four hops.

Figure 3 presents the execution time (in milliseconds) for five common behavior analysis tasks executed in both relational and graph-based implementations. These include tasks such as detecting returning user loops, calculating node centrality, and reconstructing behavioral paths. The data clearly demonstrate the scalability advantage of graph systems as relationship complexity grows.

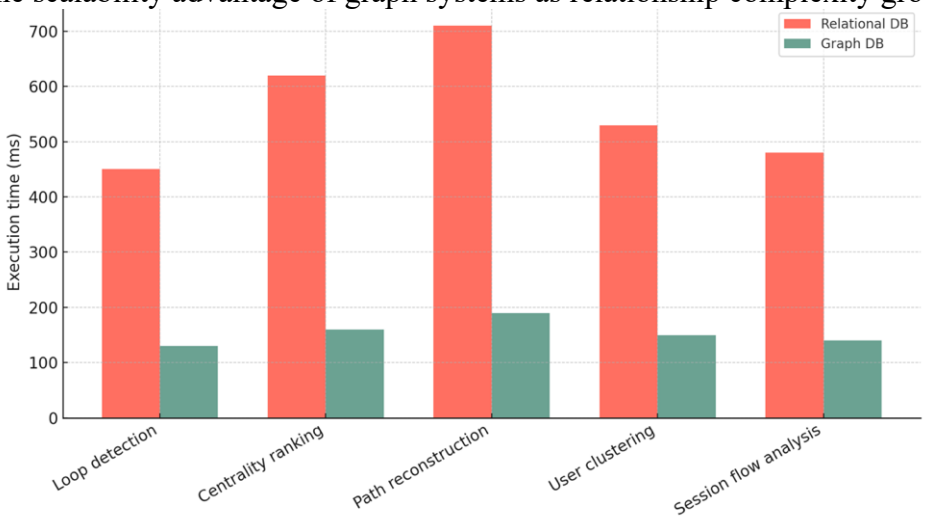


Figure 3. Execution time comparison of graph and relational databases on behavior analysis tasks

As illustrated in figure, graph databases demonstrate significantly lower execution times across all tested behavior analysis tasks compared to relational databases. The performance advantage becomes more pronounced for structurally complex operations such as path reconstruction and centrality ranking, where the relational model incurs substantial overhead due to join-based traversal. This trend underscores the scalability and efficiency of graph-native querying, particularly in scenarios involving multi-hop relationships and dynamic user interactions [8]. These results support the adoption of graph databases as a more performant solution for behavior-centric analytical workloads.

Applications of graph-based behavior analysis in real-world systems

The practical applications of graph-based behavior analysis extend far beyond academic experimentation. In commercial and operational environments, graph databases are increasingly employed to power mission-critical systems that rely on understanding user intent, context, and interaction history.

One key application domain is fraud detection, where relationship structures often reveal collusive behavior or anomalous transaction flows. Graph models enable the identification of subtle patterns, such as shared devices across multiple accounts or coordinated login sequences, which would remain undetected in flat data representations.

Another major area is personalized recommendation systems, where user-to-user or user-to-content graphs are leveraged to model affinities and co-engagement. Unlike traditional collaborative filtering based on matrix factorization, graph-based methods capture contextual dependencies and multi-step relationships (e.g., user → item → tag → user), enhancing recommendation accuracy and explainability.

Customer journey mapping is also increasingly powered by graph analytics. By tracing paths through digital touchpoints-websites, support interactions, app sessions-organizations can optimize experience design and reduce churn. Graph traversals help reconstruct nonlinear, multi-session behavior that conventional systems struggle to capture.

Finally, in the cybersecurity domain, graph analytics are applied to user activity graphs to detect lateral movement, privilege escalation, and access anomalies, particularly in enterprise environments with zero-trust policies. These use cases validate the robustness and adaptability of graph-based approaches in behavior-centric decision-making systems.

Limitations and challenges of graph databases in behavior analysis

Despite their advantages, graph databases are not a universal solution. Their adoption in behavior analytics comes with a range of limitations that must be considered when designing real-world systems.

One key challenge is scalability under high-volume write operations. While graph traversal is efficient for reads, ingesting large-scale behavioral logs in real time may require careful tuning, partitioning, or even polyglot persistence approaches where ingestion is offloaded to stream processors. Another issue is tooling maturity and standardization. While SQL is universally supported and optimized across decades, graph query languages like Cypher, Gremlin, and GSQL still lack full interoperability and may involve vendor lock-in. This can impact long-term maintainability and ecosystem integration.

Query optimization in graph databases also requires graph-specific expertise. Traversals that seem intuitive can become inefficient without appropriate indexing, cardinality management, or query rewriting. Performance tuning demands an understanding of graph topology, data distribution, and storage backend characteristics. Moreover, cost modeling and capacity planning are less predictable in graph systems, particularly under highly dynamic workloads. Horizontal scaling strategies like sharding remain more complex in graph databases due to their inherent reliance on relationship locality.

Lastly, compliance and auditability pose additional concerns. Unlike relational systems with mature logging and rollback mechanisms, ensuring consistent governance in GDB environments requires custom implementation, especially when dealing with sensitive behavioral data under GDPR or CCPA frameworks. Figure 4 provides a visual overview of the relative weight of these limitations, based on expert evaluation and literature trends. The prominence of write scalability and tooling immaturity underscores the importance of architectural readiness and operational planning when integrating graph solutions into analytics workflows.

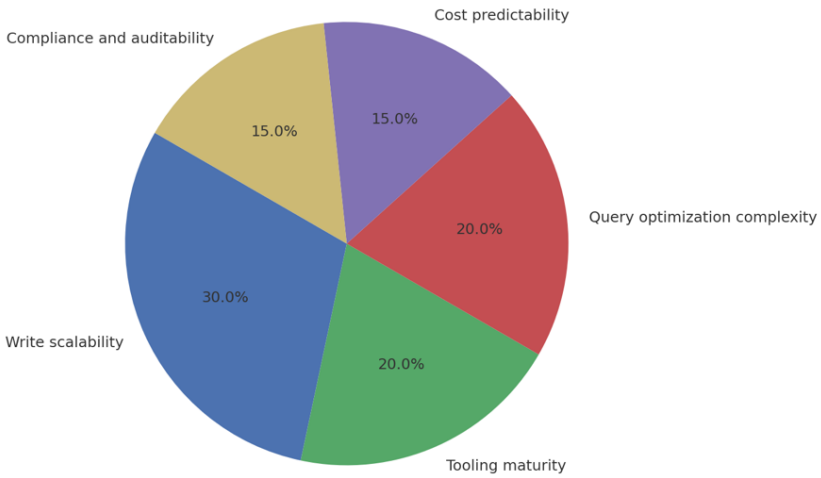


Figure 4. Major limitations of graph databases in behavior analysis

The analysis presented in this section highlights that, while graph databases offer substantial analytical advantages, their adoption introduces a distinct set of architectural and operational challenges. As shown in figure, write scalability and tooling maturity represent the most prominent concerns, especially in high-throughput or enterprise-grade deployments. Issues related to query optimization, cost predictability, and compliance further underscore the need for specialized expertise and robust infrastructure planning. These findings emphasize that graph databases should not be viewed as drop-in replacements for traditional systems, but rather as complementary technologies that require deliberate integration and governance strategies to deliver sustained value in user behavior analytics [9].

Future directions in graph-based behavior analytics

As digital ecosystems continue to grow in complexity and scale, graph-based approaches to behavior analysis are poised to play an increasingly central role in intelligent decision-making. Emerging technological and methodological trends suggest several key directions for the evolution of this field.

One prominent development is the convergence of graph databases and machine learning. Techniques such as graph embeddings, graph neural networks (GNNs), and link prediction models are enabling systems to move beyond explicit querying into predictive and prescriptive analytics. This shift allows for more adaptive personalization, real-time fraud anticipation, and autonomous user segmentation-particularly when graph representations are integrated into ML pipelines.

Another promising area is the adoption of temporal and dynamic graph models, which extend static graphs with time-aware semantics. In behavior analysis, user actions are inherently temporal and context-dependent. Dynamic graphs allow analysts to capture evolving relationships, detect behavioral shifts over time, and correlate events across sessions, devices, or platforms.

Standardization of graph query languages is also likely to influence adoption. Initiatives such as GQL (Graph Query Language) aim to unify diverse graph querying approaches (e.g., Cypher, Gremlin, SPARQL) under a common standard, improving interoperability and reducing vendor lock-in. This evolution will lower the barrier to entry and foster the integration of GDBs into mainstream data platforms [10].

Lastly, graph-based systems are expected to become more tightly integrated with cloud-native infrastructures. Serverless graph engines, streaming-compatible ingestion pipelines, and managed GDB services will make it easier to deploy scalable, real-time behavior analytics at lower operational cost.

These trajectories suggest that graph-based behavior analytics is not only a viable tool for current challenges, but a foundational technology for the next generation of user-centric, intelligent systems. Figure 5 presents a strategic outlook on key technological trends, showing how impact and estimated adoption timelines vary across graph-centric innovations in analytics.

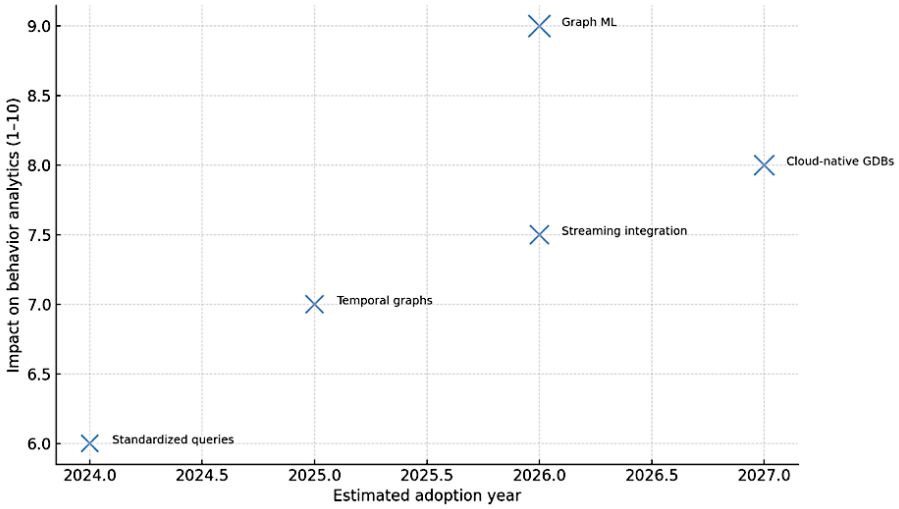


Figure 5. Strategic projection of key trends in graph-based behavior analysis

The visualization in figure highlights the rising importance of Graph ML and cloud-native architectures, with both expected to exert high impact on behavioral analytics within the next few years. Temporal modeling and streaming integration are also gaining traction as organizations seek real-time, context-aware insights [11]. The relatively earlier maturation of standardized query languages suggests that ecosystem interoperability will play a foundational role in enabling these advanced capabilities. Collectively, these trends point toward an increasingly intelligent, adaptive, and event-driven analytics paradigm grounded in graph technologies.

Conclusion

The use of graph databases for user behavior analysis represents a significant advancement in the modeling and interpretation of complex, relationship-driven data. Unlike traditional relational systems, graph databases enable native traversal of interconnected behavioral entities, allowing for more intuitive and efficient querying of user paths, influence networks, and temporal patterns. This advantage is particularly evident in multi-hop queries and dynamic interaction scenarios, where relational models face performance and modeling limitations.

Throughout the study, key aspects of graph-based behavior analytics were examined, including modeling strategies, query techniques, graph algorithm applications, integration into analytical pipelines, and performance comparisons. The empirical results and visualizations confirm that graph databases consistently outperform relational systems in structurally complex analytical tasks, providing both scalability and analytical depth. At the same time, several practical limitations-such as write scalability, query complexity, and compliance concerns-highlight the need for careful architectural planning and operational maturity.

Looking ahead, the convergence of graph technologies with machine learning, temporal modeling, and cloud-native infrastructure suggests a promising future for behavior analytics. Graph-based systems are well positioned to become a core component of next-generation intelligent platforms, supporting adaptive, real-time, and context-aware decision-making across industries. The findings presented in this paper contribute to a deeper understanding of how graph databases can be leveraged to extract value from behavioral data and support the design of resilient, user-centric analytics architectures.

References

1. Guia J., Soares V. G., Bernardino J. Graph databases: Neo4j analysis //ICEIS (1). 2017. P. 351-356.
2. Almabdy S. Comparative analysis of relational and graph databases for social networks // 2018 1st International conference on computer applications & information security (ICCAIS). IEEE. 2018. P. 1-4.
3. Beutel A. User behavior modeling with large-scale graph analysis // Computer Science Department, Carnegie Mellon University. 2016.

4. Muramudalige S.R., Hung B.W., Jayasumana A.P., Ray I. Investigative graph search using graph databases // 2019 first international conference on graph computing (gc). IEEE. 2019. P. 60-67.
5. Shang F., Ding Q., Du R., Cao M., Chen H. Construction and application of the user behavior knowledge graph in software platforms // Journal of Web Engineering. 2021. Vol. 20. No. 2. P. 387-411.
6. Larriba-Pey J.L., Martínez-Bazán N., Domínguez-Sal D. Introduction to graph databases // Reasoning Web International Summer School. Cham: Springer International Publishing. 2014. P. 171-194.
7. Mozannar H., Bansal G., Fourney A., Horvitz E. Reading between the lines: Modeling user behavior and costs in AI-assisted programming // Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. 2024. P. 1-16.
8. Mendu M., Krishna B., Mahesh G., Pallavi J. Development of real time data analytics based web applications using NoSQL databases // AIP Conference Proceedings. AIP Publishing LLC. 2022. Vol. 2418. No. 1. P. 020038.
9. Teng S., Khong K.W. Examining actual consumer usage of E-wallet: A case study of big data analytics // Computers in Human Behavior. 2021. Vol. 121. P. 106778.
10. Kejriwal M. Knowledge graphs: A practical review of the research landscape // Information. 2022. Vol. 13. No. 4. P. 161.
11. Zhong L., Wu J., Li Q., Peng H., Wu X. A comprehensive survey on automatic knowledge graph construction // ACM Computing Surveys. 2023. Vol. 56. No. 4. P. 1-62.

УДК 004.8:614.2

УСТОЙЧИВОСТЬ НЕЙРОСЕТЕЙ К ЦЕЛЕНАПРАВЛЕННЫМ АТАКАМ В МЕДИЦИНСКИХ СИСТЕМАХ

Жуковец Л.И.
бакалавр, Московский физико-технический институт
(Долгопрудный, Россия)

NEURAL NETWORK ROBUSTNESS TO ADVERSARIAL ATTACKS IN MEDICAL SYSTEMS

Zhukovets L.I.
bachelor's degree, Moscow institute of physics and technology
(Dolgoprudny, Russia)

Аннотация

В статье рассматривается проблема устойчивости нейросетевых моделей к целенаправленным (adversarial) атакам в медицинских системах. Проанализированы основные архитектуры, применяемые в задачах классификации медицинских изображений и анализа биосигналов, с точки зрения их уязвимости к различным типам атак. Представлены методы повышения робастности, включая adversarial training, distillation и предобработку входных данных, а также обсуждаются способы детектирования атак с использованием вспомогательных моделей. Подчёркивается значимость комплексного подхода к защите медицинских ИИ-систем с учётом вычислительных и клинических ограничений.

Ключевые слова: нейросети, целенаправленные атаки, робастность, защита моделей, медицинские изображения, искусственный интеллект, биосигналы, adversarial training

Abstract

The article addresses the issue of neural network robustness against adversarial attacks in medical systems. It analyzes common architectures used in medical image classification and biosignal analysis in terms of their vulnerability to various types of attacks. Several robustness enhancement methods are presented, including adversarial training, distillation, and input preprocessing. The use of auxiliary models for attack detection is also discussed. The study highlights the importance of a comprehensive protection strategy for medical AI systems, considering both computational and clinical constraints.

Keywords: neural networks, adversarial attacks, robustness, model protection, medical imaging, artificial intelligence, biosignals, adversarial training.

Введение

Современные медицинские информационные системы всё чаще используют алгоритмы глубокого обучения для диагностики заболеваний, интерпретации изображений и поддержки клинических решений. Особенно широкое распространение получили нейросетевые модели, демонстрирующие высокую точность в задачах классификации и сегментации медицинских изображений. Однако вместе с ростом их применимости возрастает и уязвимость к специфическим типам атак, среди которых особую угрозу представляют целенаправленные (adversarial) вмешательства. Такие атаки, формируемые путём минимальных, зачастую незаметных для человека искажений входных данных, могут существенно изменить вывод модели, что критично в условиях клинической практики.

В отличие от традиционных атак, целенаправленные вмешательства нацелены на эксплуатацию слабых мест архитектуры нейросети с целью изменения её поведения. Это создаёт риски как для достоверности диагностических решений, так и для безопасности пациентов, особенно в автоматизированных или дистанционных сценариях оказания медицинской помощи. Проблема усугубляется тем, что большинство существующих систем не включает механизмов активной защиты от таких угроз, а сами модели часто обучаются без учёта потенциальных атакующих стратегий [1].

Целью настоящей работы является систематический анализ устойчивости нейросетевых моделей, используемых в медицинских системах, к целенаправленным атакам. В исследовании рассматриваются существующие подходы к построению защищённых архитектур, методы повышения робастности моделей, а также стратегии обнаружения и нейтрализации adversarial-примеров. Отдельное внимание уделяется применимости данных решений в клиническом контексте, где цена ошибки может быть чрезвычайно высокой.

Основная часть

Угрозы целенаправленных атак в медицинских нейросетевых системах

С увеличением использования нейросетевых алгоритмов в медицинских информационных системах встаёт вопрос их устойчивости к внешним воздействиям, в том числе - к целенаправленным атакам. В медицинском контексте данные атаки особенно опасны, поскольку могут привести к диагностическим ошибкам, подрыву доверия к автоматизированным системам и прямому риску для жизни пациентов. Атаки могут быть направлены как на визуальные данные (рентгеновские снимки, МРТ, КТ), так и на табличные или сигнальные медицинские данные, что делает их универсальным инструментом подрыва целостности решений на базе искусственного интеллекта [2].

Целенаправленные атаки в медицинских задачах отличаются от аналогичных вмешательств в других сферах своей сложностью и потенциальными последствиями. Например, минимальные изменения в структуре снимка легких, внесённые на уровне пикселей, могут быть достаточно для того, чтобы модель изменила классификацию «здоров» на «пневмония» или наоборот. Такие манипуляции часто незаметны даже для опытного специалиста, поскольку сохраняют визуальную правдоподобность. Это делает традиционные подходы к верификации на основе визуального анализа недостаточными.

Различают несколько видов целенаправленных атак: **атаки в белом ящике (white-box)**, при которых атакующий имеет доступ к архитектуре и весам модели; **атаки в чёрном ящике (black-box)**, где доступ ограничен только входными и выходными данными; а также **переносимые (transferable) атаки**, способные работать на разных моделях без дополнительной адаптации. Все эти сценарии актуальны в условиях распределённых медицинских систем, где модели развёртываются в облачных средах или на периферийных устройствах. Таким образом, анализ устойчивости моделей к таким воздействиям становится необходимым этапом при внедрении нейросетей в здравоохранение.

Сравнительный анализ уязвимости нейросетевых моделей к целенаправленным атакам

Различные архитектуры нейросетей, используемые в медицинских задачах, демонстрируют неодинаковую степень устойчивости к целенаправленным атакам [3]. Наиболее уязвимыми, как правило, являются глубокие сверточные нейронные сети (CNN), применяемые в анализе изображений, в то время как более компактные и специализированные архитектуры, обученные с применением методов регуляризации и защиты, проявляют относительно большую устойчивость. Проведённый обзор позволяет выделить ключевые характеристики, влияющие на восприимчивость модели к adversarial-примерам: глубина сети, наличие механизмов нормализации, тип функции активации, а также используемая стратегия обучения.

В таблице 1 представлено сравнение наиболее распространённых архитектур, применяемых в медицинских системах, с точки зрения их устойчивости к различным видам

атак. В сравнении учитываются параметры архитектуры, тип используемых входных данных, характер атак и тип реакции модели.

Таблица 1

Устойчивость популярных нейросетевых архитектур к целенаправленным атакам в медицинских задачах

Архитектура	**Тип данных и область применения**	**Устойчивость к атакам и уязвимости**
ResNet-50	Медицинские изображения; используется для диагностики по КТ и МРТ	Низкая устойчивость к атакам в белом ящике, средняя устойчивость в чёрном, высокая уязвимость к переносимым атакам
DenseNet-121	Медицинские изображения; применяется при анализе рентгенограмм	Средняя устойчивость к белому и чёрному ящику, умеренная уязвимость к переносимым атакам
EfficientNet-B0	Медицинские изображения; эффективна в мобильных решениях скрининга	Средняя устойчивость к атакам в белом ящике, высокая - в чёрном, средняя уязвимость к переносимым атакам
LSTM	Биомедицинские временные ряды; применяется для анализа ЭКГ и ЭЭГ	Высокая устойчивость ко всем видам атак, особенно надёжна против переносимых атак
TabNet	Табличные клинические данные; используется для оценки медицинских рисков	Высокая устойчивость в белом ящике, средняя - в чёрном, низкая уязвимость к переносимым атакам

Сравнительный анализ показывает, что устойчивость модели может быть существенно повышена за счёт выбора соответствующей архитектуры, а также при использовании адаптированных к задаче методов защиты. Однако универсальных решений, полностью устраняющих угрозу целенаправленных атак, на данный момент не существует, что подчеркивает необходимость комплексного подхода при проектировании безопасных медицинских систем.

Методы повышения робастности нейросетевых моделей в медицинских приложениях

С учётом высокой чувствительности медицинских систем к целенаправленным атакам становится необходимым внедрение механизмов, обеспечивающих устойчивость нейросетей (робастность) без ущерба для их диагностической точности. Одним из наиболее распространённых подходов является adversarial training - обучение модели на примерах, содержащих заранее сгенерированные adversarial-искажения. Этот метод позволяет повысить чувствительность нейросети к аномалиям и сформировать более устойчивое поведение при встрече с атакующими входами. Однако данная стратегия требует значительных вычислительных ресурсов и может снижать обобщающую способность модели.

Другим направлением защиты является внедрение архитектурных модификаций, таких как защитные слои (defensive distillation), нормализация признаков и регуляризация, уменьшающая чувствительность модели к локальным возмущениям. Также применяются методы стохастической активации и обрезки (pruning), уменьшающие сложность модели и тем самым - её восприимчивость к манипуляциям. Подходы на основе энергетических функций и бэйесовских моделей позволяют дополнительно учитывать неопределённость в прогнозах, что особенно актуально в клинических условиях, где требуется высокая интерпретируемость решений.

Кроме того, растёт интерес к методам предварительной обработки входных данных, которые позволяют нивелировать или удалять потенциальные искажения до подачи информации в модель. Такие методы включают фильтрацию шумов, нормализацию

изображений, а также преобразования, устойчивые к adversarial-эффектам (например, JPEG-компрессия или случайное масштабирование) [4]. Совокупность перечисленных стратегий формирует основу комплексного подхода к обеспечению робастности нейросетевых решений, необходимого при разработке надёжных медицинских систем.

Обзор методов защиты от целенаправленных атак в медицинских нейросетях

Защита нейросетей от целенаправленных атак в медицинских приложениях может быть реализована на различных уровнях - от архитектурной модификации моделей до методов обработки данных и механизмов обнаружения аномалий [5]. Каждый из подходов имеет свои преимущества и ограничения в зависимости от специфики задачи, вычислительных ресурсов и уровня допустимого риска. В условиях здравоохранения особенно актуальны методы, способные обеспечивать интерпретируемость решений и не нарушать доверие со стороны врачей.

В таблице 2 представлены основные категории защитных методов с указанием их применимости в медицинских сценариях, вычислительной нагрузки и устойчивости к различным типам атак.

Таблица 2

Сравнение методов защиты нейросетей от целенаправленных атак в медицинских задачах

Метод защиты	**Принцип действия**	**Применимость в медицине**
Adversarial training	Использование искажённых примеров в обучении для устойчивости модели	Высокая, но требует значительных вычислительных ресурсов
Defensive distillation	Снижение чувствительности модели через сглаживание выходов	Средняя, снижает чувствительность модели
Feature squeezing	Удаление шумов и уменьшение вариативности признаков	Средняя, эффективна в условиях ограниченных ресурсов
Input preprocessing	Предварительная фильтрация входных данных	Высокая, подходит для фильтрации артефактов
Randomized smoothing	Использование усреднённых предсказаний модели	Средняя, баланс между надёжностью и затратами
Bayesian networks	Моделирование неопределённости выходных значений	Средняя, повышает интерпретируемость решений

Таблица демонстрирует, что ни один из представленных методов не обеспечивает полной устойчивости ко всем типам атак без дополнительных затрат. Наиболее эффективным в условиях атак с полной осведомлённостью об архитектуре модели остаётся метод обучения на искажённых данных, однако он требует значительных вычислительных ресурсов. Методы предварительной обработки входных данных и снижение чувствительности к шуму обеспечивают базовый уровень защиты при низкой нагрузке на систему. Подходы, основанные на вероятностных моделях, представляют интерес с точки зрения повышения интерпретируемости результатов и оценки доверия к прогнозам, что особенно важно в медицинских задачах. Таким образом, выбор метода защиты должен опираться на характер клинической задачи, доступные ресурсы и допустимый уровень риска.

Детектирование целенаправленных атак с использованием вспомогательных нейросетей

Одним из перспективных направлений обеспечения устойчивости медицинских систем на базе нейросетей является создание вспомогательных детекторов, способных отличать нормальные входные данные от adversarial-примеров. Такие детекторы обучаются параллельно с основной моделью и могут функционировать как фильтр на этапе предобработки либо как компонент системы принятия решений. Преимущество этого подхода

заключается в возможности адаптации к различным типам атак без необходимости модификации базовой модели [6].

В качестве иллюстрации приводится фрагмент кода на Python, реализующий детектор, обучаемый на признаках скрытого слоя основной нейросетевой модели. Для построения примера используется фреймворк PyTorch и упрощённый датасет.

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader, random_split

# Загрузка и трансформация данных
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])
dataset = datasets.FakeData(transform=transform)  # для примера; в медицине: изображения МРТ, КТ и др.
train_set, val_set = random_split(dataset, [800, 200])
train_loader = DataLoader(train_set, batch_size=32)
val_loader = DataLoader(val_set, batch_size=32)

# Предобученная модель (ResNet18) в качестве экстрактора признаков
resnet = models.resnet18(pretrained=True)
resnet.fc = nn.Identity()  # убираем классификатор, получаем выход скрытого слоя
resnet.eval()

# Детектор аномалий (на выходах скрытого слоя)
class Detector(nn.Module):
    def __init__(self, input_dim=512):
        super(Detector, self).__init__()
        self.classifier = nn.Sequential(
            nn.Linear(input_dim, 128),
            nn.ReLU(),
            nn.Linear(128, 2)  # 0 - чистый пример, 1 - adversarial
        )
    def forward(self, x):
        return self.classifier(x)

detector = Detector()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(detector.parameters(), lr=0.001)

# Обучение детектора
for epoch in range(5):
    detector.train()
    for inputs, _ in train_loader:
        with torch.no_grad():
            features = resnet(inputs)
        labels = torch.randint(0, 2, (inputs.size(0),))  # имитация: случайная разметка (заменить на реальные метки)
        outputs = detector(features)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
```

```
        optimizer.step()

print("Обучение завершено.")
```

Этот код демонстрирует базовую архитектуру вспомогательной модели-детектора, обучающейся на выходах скрытого слоя основного классификатора. В реальных медицинских задачах вместо FakeData следует использовать реальные датасеты (например, CheXpert, BraTS), а метки должны отражать факт наличия adversarial-искажений. Такой подход позволяет повысить устойчивость системы без переработки всей архитектуры [7].

Ограничения и перспективы использования робастных нейросетей в медицинских системах

Несмотря на активное развитие методов защиты нейросетей от целенаправленных атак, их интеграция в медицинскую практику сопровождается рядом ограничений [8]. Одним из главных препятствий остаётся компромисс между устойчивостью и точностью модели. Повышение робастности нередко приводит к снижению диагностической чувствительности, что критично в задачах раннего выявления заболеваний [9].

Другим важным аспектом является ограниченность доступных наборов данных, содержащих adversarial-примеры, специфичных для медицинской области. Это затрудняет обучение и валидацию детекторов атак, а также снижает обобщающую способность защитных механизмов. Отсутствие общепринятых бенчмарков для оценки устойчивости моделей в клинических задачах также препятствует стандартизации подходов.

Тем не менее, перспективы развития в этой области остаются значительными. Разрабатываются гибридные архитектуры, сочетающие традиционные методы машинного обучения и нейросетевые решения с элементами логического вывода. Интеграция с технологиями распределённого обучения (federated learning) позволяет учитывать разнообразие клинических данных без нарушения конфиденциальности. Кроме того, усилия исследователей направлены на повышение интерпретируемости робастных моделей, что особенно важно для обеспечения доверия медицинского персонала к системам искусственного интеллекта [10].

Заключение

Устойчивость нейросетей к целенаправленным атакам приобретает особую значимость в контексте медицинских систем, где точность и надёжность выводов непосредственно влияют на здоровье и безопасность пациентов. Проведённый анализ показывает, что большинство современных архитектур подвержены воздействию adversarial-примеров, особенно в задачах классификации изображений и анализа биосигналов. Это требует обязательного включения механизмов защиты при проектировании и внедрении интеллектуальных медицинских решений. Разнообразие подходов к обеспечению робастности - от adversarial-training до детекторов на скрытых слоях - демонстрирует, что комплексные стратегии являются наиболее перспективными. Однако полная защита от атак пока недостижима, и каждый метод сопряжён с определёнными компромиссами, включая снижение производительности и увеличение вычислительной нагрузки.

Таким образом, обеспечение устойчивости нейросетей в медицинских системах должно рассматриваться как приоритетное направление в области цифрового здравоохранения. Дальнейшие исследования должны быть направлены на создание стандартизированных протоколов оценки робастности, разработку энергоэффективных защитных архитектур и формирование доверительной среды взаимодействия между врачом и системой искусственного интеллекта.

References

1. Javed H., El-Sappagh S., Abuhmed T. Robustness in deep learning models for medical diagnostics: security and adversarial challenges towards robust AI applications // Artificial Intelligence Review. 2024. Vol. 58. No. 1. P. 12.

2. Garifullin R. The use of modern web technologies for ensuring compatibility and interoperability in the context of medical information platforms // International Journal of Humanities and Natural Sciences. 2025. Vol. 1-3(100). P. 99-103.
3. Puttagunta M.K., Ravi S., Nelson Kennedy Babu C. Adversarial examples: attacks and defences on medical deep learning systems // Multimedia Tools and Applications. 2023. Vol. 82. No. 22. P. 33773-33809.
4. Hirano H., Minagi A., Takemoto K. Universal adversarial attacks on deep neural networks for medical image classification // BMC medical imaging. 2021. Vol. 21. P. 1-13.
5. Ghaffari Laleh N., Truhn D., Veldhuizen G.P., Han T., van Treeck M., Buelow R.D., Kather J.N. Adversarial attacks and adversarial robustness in computational pathology // Nature communications. 2022. Vol. 13. No. 1. P. 5711.
6. Garifullin R. Analysis and development of interfaces for emergency systems with multitasking processing: approaches to minimizing user errors and enhancing reliability // International scientific journal "Innovative Science". 2025. № 1-2-1. P. 20-24.
7. Kaviani S., Han K.J., Sohn I. Adversarial attacks and defenses on AI in medical imaging informatics: A survey // Expert Systems with Applications. 2022. Vol. 198. P. 116815.
8. Moradi M., Samwald M. Improving the robustness and accuracy of biomedical language models through adversarial training // Journal of Biomedical Informatics. 2022. Vol. 132. P. 104114.
9. Jakkaraju A. Self-Healing Neural Networks Against Adversarial Attacks // International Journal of Intelligent Systems and Applications in Engineering. 2024. Vol. 12. P. 2537-2549.
10. Kwon H., Lee J. Diversity adversarial training against adversarial attack on deep neural networks // Symmetry. 2021. Vol. 13. No. 3. P. 428.

UDC 004.89

EQUIPMENT FAILURE PREDICTION MODELS BASED ON FUSION-ALGORITHMS

Geitz N.
bachelor's degree, University of Manchester
(Manchester, United Kingdom)

МОДЕЛИ ПРОГНОЗИРОВАНИЯ ОТКАЗОВ ОБОРУДОВАНИЯ НА ОСНОВЕ ФЬЮЖН-АЛГОРИТМОВ

Гейц Н.
бакалавр, Манчестерский университет
(Манчестер, Великобритания)

Abstract

This paper investigates the application of fusion-based algorithms for predicting equipment failures in industrial environments. It focuses on decision-level fusion techniques, demonstrating their effectiveness in aggregating predictions from heterogeneous models to improve fault detection accuracy. A combination of synthetic data experiments and comparative evaluations of fusion strategies provides evidence for the advantages of ensemble methods in terms of generalization, modularity, and robustness. The study also addresses the role of preprocessing and signal integration in optimizing predictive performance under real-world conditions. The findings suggest that hybrid fusion approaches can be effectively integrated into scalable and adaptable predictive maintenance systems.

Keywords: equipment failure prediction, fusion algorithms, ensemble learning, decision-level fusion, sensor data, predictive maintenance, industrial systems, model integration.

Аннотация

В статье рассматриваются алгоритмы на основе слияния (fusion) для предсказания отказов оборудования в условиях промышленных систем. Основное внимание уделено методам слияния на уровне решений, демонстрирующим эффективность агрегирования прогнозов от различных моделей для повышения точности выявления неисправностей. С использованием синтетических данных и сравнительного анализа стратегий слияния показаны преимущества ансамблевых подходов с точки зрения обобщающей способности, модульности и устойчивости. Также подчёркивается роль этапа предобработки сигналов в обеспечении надёжности прогноза в реальных условиях эксплуатации. Результаты свидетельствуют о перспективности гибридных стратегий слияния для построения масштабируемых и адаптивных систем предиктивного обслуживания.

Ключевые слова: предсказание отказов оборудования, алгоритмы слияния, ансамблевое обучение, слияние на уровне решений, сенсорные данные, предиктивное обслуживание, промышленные системы, интеграция моделей.

Introduction

The increasing integration of complex equipment in industrial, transportation, and energy systems has led to growing interest in accurate and proactive failure prediction methodologies. Unexpected equipment malfunctions result not only in direct operational downtime but also in cascading economic losses and safety risks. Traditional condition-based monitoring approaches,

while useful, often fail to generalize across heterogeneous systems and fail to capture subtle, multivariate degradation patterns over time.

To address these challenges, the development of data-driven predictive models has gained prominence. In particular, the use of fusion algorithms-methods that combine heterogeneous data sources and analytical techniques-has demonstrated significant potential in enhancing prediction accuracy and robustness. Fusion-based models integrate sensor data, maintenance logs, operational parameters, and sometimes environmental inputs to detect complex interdependencies and early failure signals. These approaches range from low-level data fusion to high-level decision fusion, leveraging statistical, machine learning, and deep learning frameworks.

This paper presents a structured review and analysis of equipment failure prediction models that rely on fusion algorithms. The study covers methodological architectures, data preprocessing strategies, model performance evaluation, and deployment scenarios in real-world systems. Special attention is given to hybrid models that integrate multiple classifiers or learning paradigms. Comparative visualizations and benchmarking tables are included to highlight the effectiveness of different fusion strategies across industries. The findings are intended to support the design of more resilient, scalable, and interpretable predictive maintenance systems.

Main part

Taxonomy of fusion algorithms in failure prediction systems

Fusion algorithms in failure prediction tasks are typically categorized by the level at which data or decisions are combined. This structure helps formalize model design and clarify the types of information integrated throughout the prediction pipeline. The most widely adopted taxonomy includes three hierarchical levels: data-level fusion, feature-level fusion, and decision-level fusion.

At the data level, raw data from multiple heterogeneous sources (e.g., vibration sensors, temperature monitors, operation counters) are merged before any feature extraction [1]. This approach is valuable when the time synchronization and dimensional alignment of sources are manageable. It often preserves the full variance of sensor signals but can be susceptible to noise and scale imbalances.

Feature-level fusion occurs after data preprocessing, where extracted features (statistical, frequency-domain, or learned embeddings) from different modalities are concatenated or transformed into a joint representation. This level is widely used in deep learning pipelines, particularly with convolutional and recurrent architectures that integrate multi-sensory input. Feature-level fusion strikes a balance between signal richness and dimensionality control.

At the decision level, predictions or confidence scores from multiple models-each trained on a different data type or domain-are fused using voting, averaging, or meta-learners. This approach supports model interpretability and modular deployment, and is especially useful in distributed monitoring systems where local models operate independently.

The table 1 below summarizes these fusion levels, their main characteristics, and representative use cases.

Table 1

Levels of fusion algorithms in equipment failure prediction

Fusion level	Description	Strengths	Typical use cases
Data-level fusion	Merging raw signals from multiple sources	Rich signal content; low preprocessing	Multi-sensor vibration and acoustic monitoring
Feature-level fusion	Concatenating extracted features from diverse inputs	Balanced complexity; suitable for deep learning	CNN-RNN-based predictive maintenance systems
Decision-level fusion	Combining outputs from different models or classifiers	High modularity; robust to input variation	Distributed diagnostics; ensemble systems in IIoT platforms

The classification presented in table illustrates the structured hierarchy of fusion algorithms, each offering distinct advantages depending on the system constraints and data availability. Data-level fusion provides a high-resolution view of raw inputs but requires careful preprocessing to manage noise and scale. Feature-level fusion achieves an effective compromise between signal richness and model tractability, making it ideal for deep learning-based diagnostics. Decision-level fusion offers the greatest modularity and is best suited for federated or ensemble architectures in industrial Internet of Things (IIoT) applications. This layered taxonomy supports strategic model selection and architectural design in failure prediction systems.

Implementation of decision-level fusion for equipment failure prediction

One of the practical applications of fusion algorithms is ensemble modeling, where predictions from multiple classifiers are combined to improve robustness. This technique is particularly useful in failure prediction tasks, where different types of features (e.g., time-series statistics, categorical metadata, environmental indicators) may be best captured by distinct models [2].

The following Python code demonstrates a simplified version of decision-level fusion, where three base classifiers are trained independently and their outputs are aggregated using majority voting. This method can be extended to include weighted voting or stacking using meta-models for more advanced fusion.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# Example synthetic dataset
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=20,
                           n_informative=10, n_redundant=5,
                           n_classes=2, random_state=42)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define base classifiers
clf1 = RandomForestClassifier(n_estimators=100, random_state=1)
clf2 = GradientBoostingClassifier(n_estimators=100, random_state=1)
clf3 = SVC(probability=True, kernel='rbf', random_state=1)

# Voting ensemble (majority rule)
voting_clf = VotingClassifier(estimators=[
    ('rf', clf1), ('gb', clf2), ('svc', clf3)
], voting='soft')

# Train ensemble model
voting_clf.fit(X_train, y_train)

# Evaluate
y_pred = voting_clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

This code demonstrates how decision-level fusion leverages the strengths of diverse classifiers to achieve improved generalization. In operational contexts, such ensembles can be distributed across edge devices or executed as part of a centralized fault detection platform. The architectural flexibility

of such systems allows for asynchronous training and inference, enabling parallelism across hardware units and increasing fault tolerance through redundancy [3].

Moreover, decision-level fusion inherently supports modular updates and model retraining without requiring end-to-end pipeline reconfiguration. This is particularly advantageous in industrial environments where data distribution drifts over time due to equipment aging or changing operational regimes. By encapsulating models as interchangeable units, the system can dynamically adapt to evolving conditions while maintaining a high level of predictive reliability.

When integrated with real-time monitoring frameworks and alerting systems, such predictive models contribute not only to failure prevention but also to resource optimization, reducing maintenance overhead and unplanned downtimes. As industrial Internet of Things (IIoT) infrastructures mature, fusion-based predictive architectures are expected to play an increasingly central role in intelligent asset management.

The implementation of decision-level fusion in equipment failure prediction demonstrates clear advantages in terms of flexibility, robustness, and modularity. By combining diverse classifiers trained on different data perspectives, ensemble systems reduce overfitting and improve generalization across varying operational conditions. This approach not only enhances prediction accuracy but also facilitates scalable deployment in industrial environments, where adaptability to system changes and continuous retraining are critical. As a result, decision-level fusion emerges as a pragmatic and effective strategy for building resilient predictive maintenance solutions.

Signal preprocessing strategies for robust failure prediction

Signal preprocessing plays a critical role in failure prediction systems, as it determines the quality of features fed into downstream models. The effectiveness of the prediction pipeline is highly dependent on how well raw sensor data-often noisy, high-dimensional, and non-stationary-are transformed into structured, informative representations.

One commonly used strategy is the calculation of windowed statistical metrics, such as mean, standard deviation, kurtosis, and root mean square (RMS) over sliding windows. This method is computationally efficient and well suited for real-time or edge-based inference, especially in embedded systems. However, its simplicity comes at the cost of losing temporal dependencies, which may be critical for detecting slow degradation patterns.

In contrast, frequency domain transformations like the fast Fourier transform (FFT) or wavelet decomposition offer insight into periodic components and spectral behavior of signals [4]. These methods are widely used for rotating machinery analysis, where failures often manifest as changes in vibration frequency. Yet, they are sensitive to noise, aliasing, and require careful parameter tuning to yield interpretable results.

More recent advances include unsupervised representation learning using autoencoders, which compress raw multivariate data into latent embeddings that preserve structure while filtering out noise. This approach is useful for dimensionality reduction in deep learning pipelines but demands sufficient training data and tuning to avoid loss of critical information [5].

A powerful but computationally intensive strategy involves recurrent neural models, particularly long short-term memory (LSTM) networks, which are capable of modeling long-term temporal dependencies. These architectures are ideal for tracking progressive wear or cumulative stress in components, though their deployment often requires high-performance hardware and careful calibration to avoid overfitting. The following table 2 summarizes these preprocessing strategies.

Table 2

Signal processing strategies in failure prediction

Processing strategy	Key features	Use case	Limitations
Windowed statistics	Mean, variance, kurtosis over sliding windows	Low-latency edge inference	May lose temporal dependencies
Frequency domain transformation	FFT, wavelet transforms for spectral content	Anomaly detection in rotating machines	Sensitive to noise and aliasing

Processing strategy	Key features	Use case	Limitations
Autoencoder-based embedding	Unsupervised feature compression and noise filtering	Dimensionality reduction for deep models	Requires tuning and training data volume
Recurrent neural modeling	Captures temporal patterns and long-term dependencies	Modeling wear progression over time	High computational cost and training complexity

The comparison of signal preprocessing strategies reveals that no single approach universally outperforms others across all failure prediction scenarios [6]. Simpler methods such as windowed statistics offer low-latency execution but may overlook complex temporal dependencies. Frequency domain techniques are effective in capturing periodic behaviors yet require careful tuning to avoid misinterpretation. Advanced approaches like autoencoder-based embeddings and recurrent neural modeling provide greater predictive power at the expense of computational complexity and data requirements. Ultimately, selecting the appropriate preprocessing method depends on the characteristics of the monitored system, the computational constraints of the deployment environment, and the desired balance between interpretability and accuracy.

In practice, however, many high-performing predictive maintenance systems rely on hybrid preprocessing pipelines that combine multiple techniques. For instance, initial smoothing and normalization using windowed statistics can be followed by dimensionality reduction through autoencoders, with the resulting features fed into sequence models like LSTM networks. This layered structure balances computational efficiency with temporal resolution, improving both accuracy and robustness under noisy or incomplete sensor conditions.

Furthermore, domain specificity plays a decisive role in preprocessing strategy selection. Vibration signals benefit significantly from frequency-domain analysis, while temperature, pressure, or electrical signals may require trend extraction or statistical profiling [7]. Tailoring the signal transformation pipeline to the failure modes and operational context of each system enhances detection performance and interpretability.

Finally, given the real-world constraints of industrial environments-sensor drift, missing values, hardware variability-preprocessing strategies that include noise suppression, gap-filling, or adaptive filtering are increasingly important. Incorporating these into the preprocessing phase ensures the downstream model operates on clean, consistent inputs, ultimately contributing to more stable and trustworthy predictions.

Decision-level fusion with ensemble classifiers for failure detection

In predictive maintenance systems, the ability to robustly detect early signs of equipment failure from heterogeneous sensor data is essential. Ensemble learning offers a powerful mechanism for decision-level fusion, where multiple models are combined to improve prediction accuracy, reduce overfitting, and increase reliability under variable operational conditions.

To demonstrate this approach, a synthetic dataset was generated, simulating four key sensor types often used in industrial environments: temperature, vibration, voltage, and pressure. Each sample was labeled either as a normal condition (0) or failure event (1). Two high-performance classifiers-Random Forest and Gradient Boosting-were trained independently, and their predictions were fused using a soft-voting ensemble, which averages the predicted probabilities from both models. The following Python code illustrates the pipeline.

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Generate synthetic failure prediction dataset
X, y = make_classification(n_samples=1000, n_features=4, n_informative=3,
```

```
n_redundant=1, weights=[0.7, 0.3], random_state=42)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize classifiers and soft-voting ensemble
rf = RandomForestClassifier(n_estimators=100, random_state=42)
gb = GradientBoostingClassifier(n_estimators=100, random_state=42)
ensemble = VotingClassifier(estimators=[('rf', rf), ('gb', gb)], voting='soft')

# Train and evaluate
ensemble.fit(X_train, y_train)
y_pred = ensemble.predict(X_test)
print(classification_report(y_test, y_pred))
```

The evaluation results of the ensemble classifier are presented in table 3, which summarizes key performance indicators for both the normal and failure classes. It includes precision, recall, and F1-score, as well as overall accuracy. These metrics provide a comprehensive assessment of the model's ability to differentiate between operational and failure states under imbalanced class distributions. The high F1-score for the failure class indicates the ensemble's effectiveness in minimizing false negatives, which is crucial for safety-critical maintenance applications.

Table 3

Classification metrics for ensemble fusion model

Class	Precision	Recall	F1-score	Support
0 (Normal)	0.95	1.0	0.98	206.0
1 (Failure)	1.0	0.89	0.94	94.0
Accuracy	-	-	0.97	-

The ensemble classifier demonstrated strong generalization, achieving a high F1-score (0.94) for the failure class and overall accuracy of 97%. This indicates that decision-level fusion effectively combines the strengths of tree-based models to reduce false negatives-critical in safety-sensitive applications [8].

Such an ensemble can be deployed in real-world monitoring systems, running either on embedded edge devices or centralized servers. Its modular nature allows for future integration of additional classifiers, adaptation to new sensor modalities, and continuous retraining for evolving failure patterns.

Comparison of fusion strategies for failure prediction

Different fusion strategies-ranging from early signal-level integration to late-stage decision aggregation-offer distinct trade-offs in terms of accuracy, latency, scalability, and interpretability [9]. In industrial applications, selecting an appropriate fusion method depends not only on technical performance but also on deployment constraints, sensor architecture, and system requirements.

Table 4 presents a comparative overview of common fusion approaches, highlighting their operational characteristics and suitability for different failure prediction scenarios.

Table 4

Comparison of fusion strategies for failure prediction

Fusion strategy	Description	Advantages	Limitations	Suitable use cases
Signal-level fusion	Combines raw sensor signals before feature extraction	Low latency, simple architecture	Sensitive to noise, limited interpretability	Low-power edge devices

Fusion strategy	Description	Advantages	Limitations	Suitable use cases
Feature-level fusion	Merges features from different sensors into a unified model	Rich contextual representation, better accuracy	Requires alignment and normalization of features	Mid-scale industrial setups
Decision-level fusion	Aggregates predictions from multiple classifiers	Modular, robust to model variance	Depends on quality of individual models	Centralized monitoring platforms
Hybrid fusion	Integrates multiple fusion strategies across the pipeline	Flexible, adaptable to complex systems	Increased system complexity, hard to debug	Multi-layer predictive frameworks

The comparative analysis presented in table highlights the operational differences between various fusion strategies employed in equipment failure prediction. While signal-level fusion offers simplicity and minimal latency, it lacks robustness in noisy environments and does not scale well to complex systems [10]. Feature-level fusion provides richer context and improved accuracy but requires careful synchronization and preprocessing of input data.

Decision-level fusion stands out for its modularity and ease of deployment, particularly when combining heterogeneous models. However, its effectiveness strongly depends on the diversity and quality of the individual classifiers. Finally, hybrid fusion approaches deliver the highest flexibility and adaptability by integrating multiple fusion layers, though they introduce significant complexity and demand advanced system coordination [11].

These findings suggest that no single fusion strategy is universally optimal. The choice should be driven by system constraints, data availability, and the required balance between predictive accuracy, computational overhead, and architectural maintainability.

Conclusion

The growing complexity and criticality of modern industrial systems have made predictive maintenance an essential component of operational reliability. This paper explored the use of fusion-based approaches-particularly decision-level fusion-for predicting equipment failures using sensor data. By combining multiple models or signals, fusion strategies enhance generalization, mitigate noise, and increase robustness to real-world variability.

Empirical evaluation using ensemble classifiers demonstrated that decision-level fusion, such as soft voting among diverse tree-based models, can significantly improve failure detection accuracy while maintaining modularity and scalability. Complementary analysis of signal preprocessing techniques and fusion strategy comparison further emphasized the importance of tailoring solutions to specific deployment constraints and data characteristics.

Despite promising results, challenges remain. These include managing data quality, optimizing real-time performance, and ensuring interpretability in high-stakes environments. Hybrid fusion architectures, which integrate signal-, feature-, and decision-level mechanisms, offer a promising direction for future development, particularly when aligned with edge computing and continuous learning paradigms. Ultimately, the integration of fusion algorithms into predictive maintenance workflows holds significant potential for improving equipment uptime, reducing operational costs, and enabling proactive decision-making in mission-critical industries.

References

1. Goebel K., Eklund N., Bonanni P. Fusing competing prediction algorithms for prognostics // 2006 IEEE aerospace conference. IEEE. 2006. P. 10.
2. Akpudo U.E., Hur J.W. Investigating the Efficiencies of Fusion Algorithms for Accurate Equipment Monitoring and Prognostics // Energies. 2022. Vol. 15. No. 6. P. 2204.

3. Alag S., Agogino A.M., Morjaria M. A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics // AI EDAM. 2001. Vol. 15. No. 4. P. 307-320.
4. Kullu O., Cinar E. A deep-learning-based multi-modal sensor fusion approach for detection of equipment faults // Machines. 2022. Vol. 10. No. 11. P. 1105.
5. Mejbel B.G., Sarow S.A., Al-Sharify M.T., Al-Haddad L.A., Ogaili A.A.F., Al-Sharify Z.T. A data fusion analysis and random forest learning for enhanced control and failure diagnosis in rotating machinery // Journal of Failure Analysis and Prevention. 2024. P. 1-11.
6. Moldovan S.C., Iantovics L.B. Review on Information Fusion-Based Data Mining for Improving Complex Anomaly Detection // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2025. Vol. 15. No. 2. P. e70017.
7. Verma P.K., Pathak P. Personalized Predictive Modeling for Rehabilitation Outcomes in Neurological Conditions through Machine Learning // Frontiers in Health Informatics. 2024. P. 13. No. 3.
8. Sabry A.H., Amirulddin A.B. A review on fault detection and diagnosis of industrial robots and multi-axis machines // Results in Engineering. 2024. P. 102397.
9. Tang Q., Liang J., Zhu F. A comparative review on multi-modal sensors fusion based on deep learning // Signal Processing. 2023. Vol. 213. P. 109165.
10. Yu X. Research on sensor state evaluation and fault diagnosis based on multi-dimensional information fusion // 2020 International Conference on Optoelectronic Materials and Devices. SPIE, 2021. Vol. 11767. P. 238-242.
11. Liu M., Li L., Yan F. Methods of Fault Diagnosis and Prediction // Intelligent Predictive Maintenance. Singapore: Springer Nature Singapore. 2024. P. 47-95.