- **Title:** Graph-Based AI System for Real-Time Detection and Rollback of Performance Regressions in Software Deployments

- **Author:** Piyushkumar Patel, Microsoft Research (USA)

ABSTRACT

This paper presents a Graph-Based AI System leveraging Graph Neural Networks (GNNs) for real-time detection and rollback of performance regressions in software deployments. By modeling microservices and their interdependencies as dynamic temporal graphs, this system detects anomalies in real time and initiates automated rollback procedures. The integration with CI/CD pipelines and orchestration platforms like Kubernetes ensures scalability and rapid recovery, making it a transformative approach for maintaining software reliability.

---

1 INTRODUCTION

Modern software deployment pipelines are increasingly reliant on microservices architectures, which, while offering scalability and modularity, introduce significant challenges in maintaining system reliability. One of the most pressing issues is the occurrence of performance regressions during updates, where even minor changes can cascade into widespread service disruptions [9]. Traditional monitoring systems, which rely on static thresholds to detect anomalies, are often inadequate in such dynamic environments. These systems fail to account for the intricate interdependencies between microservices and hardware components, leading to delayed detection of issues and prolonged recovery times. The limitations of static threshold-based approaches underscore the need for more adaptive and intelligent solutions capable of real-time anomaly detection and automated rollback mechanisms. To address these challenges, graph-based artificial intelligence (AI) systems leveraging Graph Neural Networks (GNNs) have emerged as a promising solution. These systems are designed to monitor and analyze the complex interactions within microservices architectures by representing them as dynamic temporal graphs. The core

innovation lies in the ability of GNNs to capture both structural and temporal dependencies, enabling the detection of anomalies that traditional methods might overlook [3]. For instance, models like DOMINANT and Dual-SVDAE reconstruct graph structures and nodal attributes to identify global and structural anomalies, while AddGraph and DynAD integrate temporal information to model evolving patterns in real-time systems [1]. By incorporating these advanced techniques, graph-based AI systems provide a robust framework for detecting

performance regressions and triggering automated rollbacks with minimal human intervention. The architecture of a graph-based AI system for real-time detection and rollback typically consists of three key components: integration with the CI/CD pipeline, a GNN engine, and a rollback controller. As depicted in FIG. 1, the CI/CD pipeline serves as the entry point for deploying new software versions, where monitoring agents collect real-time metrics such as CPU usage, memory consumption, and network latency [9]. These metrics are then fed into the GNN engine, which processes the data using dynamic temporal graphs to represent microservices and their interdependencies. The rollback controller, informed by the GNN engine's anomaly scores, evaluates the severity of detected issues and executes rollback strategies accordingly. This seamless integration ensures that performance regressions are identified and mitigated before they impact end users. A critical aspect of this approach is the use of dynamic temporal graphs to model microservices and hardware dependencies. Unlike static graphs, which capture relationships at a single point in time, dynamic temporal graphs evolve to reflect changes in the system's state over time. This capability isparticularly valuable in capturing the transient nature of microservices interactions, where dependencies can shift rapidly due to scaling events or traffic fluctuations [5]. For example, the MTDGraph method employs transfer entropy to quantify causal relationships between nodes, similar to how influence weights are calculated in microservice architectures [6]. By integrating historical data and real-time observations, dynamic temporal graphs provide a comprehensive view of the system's behavior, enabling more accurate anomaly detection and prioritization of rollback actions. Compared to conventional methods, graph-based AI systems offer several distinct advantages. Traditional rollback strategies, such as blue-green deployments and feature flags, rely heavily on pre-defined configurations and manual interventions, which can be time-consuming and error-prone [9]. In contrast, graph-based systems leverage unsupervised learning techniques to adaptively

detect anomalies without requiring extensive labeled data. For instance, the EH-GAM-EGAN model combines graph attention mechanisms with Long Short-Term Memory (LSTM) networks to capture complex node relationships, achieving significant improvements in precision, recall, and F1 scores [3]. Furthermore, the scalability of GNN-based approaches ensures that they can handle large-scale deployments without compromising detection accuracy, making them suitable for modern cloud computing environments.

In summary, graph-based AI systems represent a transformative advancement in the field of real-time anomaly detection and rollback for software deployments. By addressing the limitations of traditional static threshold-based monitoring systems, these systems provide a more adaptive and intelligent approach to maintaining system reliability. The integration of dynamic temporal graphs, GNN engines, and automated rollback controllers offers a comprehensive solution for detecting performance regressions and mitigating their impact in complex microservices architectures. As this field continues to evolve, further research is needed to explore optimization techniques, enhance explainability, and develop industry benchmarks for prioritization criteria during multi-service rollbacks [1, 3, 5, 6].

---

2 SYSTEM ARCHITECTURE

The system architecture of the GNN-based monitoring system is designed to facilitate real-time anomaly detection, root cause analysis, and automated rollback mechanisms in distributed environments. As depicted in FIG. 1, the architecture comprises several critical components: the deployment monitor, graph constructor module, GNN inference engine, root cause analyzer, and rollback controller [10, 13]. Each component plays a distinct role in ensuring seamless operation, scalability, and fault tolerance within the system. The deployment monitor acts as the primary interface for collecting runtime metrics from various services. It interacts closely with the graph constructor module, which transforms these metrics into a dynamic temporal graph representing service interdependencies. This graph serves as input to the GNN inference engine, which leverages graph neural networks (GNNs) to identify anomalous patterns and predict potential failures. Upon detecting an anomaly, the root cause analyzer isolates the underlying issue by tracing dependencies across nodes, while the rollback controller

executes recovery strategies such as blue-green deployments or feature flag toggling.Monitoring agents embedded within the system are responsible for gathering real-time metrics essential for constructing the temporal graph. These metrics include CPU usage, memory consumption, query latency, and other performance indicators that reflect the health of individual services. For instance, feature flags—highlighted in Task 3 learnings—enable granular control over specific functionalities, allowing problematic features to be disabled without redeploying the entire application [9]. This capability significantly reduces downtime and enhances the precision of anomaly detection. Furthermore, monitoring agents integrate seamlessly with tools like Prometheus and Grafana, providing comprehensive visibility into system behavior. Such integration ensures that anomalies are detected promptly and actionable insights are generated to guide subsequent interventions. The workflow of the GNN-based monitoring system can be further elucidated through the flowchart presented in FIG. 2. Initially, monitoring agents collect raw data from deployed services, which is then processed by the graph constructor module to create a structured representation of service interactions. This graph captures both static dependencies (e.g., API calls between microservices) and dynamic behaviors (e.g., variations in response times). Once constructed, the graph is fed into the GNN inference engine, where advanced algorithms analyze node embeddings to detect deviations from normal operational patterns. If an anomaly is identified, the root cause analyzer evaluates the affected nodes and their relationships to pinpoint the source of the issue. Based on the severity and impact score, the rollback controller determines whether a partial rollback (e.g., disabling a feature flag) or a full rollback (e.g., reverting to a previous deployment version) is necessary. This structured approach ensures rapid recovery and minimizes disruptions to end users. KServe and Kubernetes play pivotal roles in optimizing resource allocation and enabling scalable deployments within the GNN-based monitoring system. KServe, as a de facto standard for model serving on Kubernetes, provides capabilities such as autoscaling, canary deployments, and multi-model serving [15]. Autoscaling allows the system to dynamically adjust resource allocation based on demand, scaling from zero pods during idle periods to multiple pods during peak loads. This feature not only reduces infrastructure costs but also ensures consistent performance under varying workloads. Additionally, Kubernetes' orchestration capabilities distribute tasks efficiently across services, preventing single points of failure and enhancing resilience against traffic surges. The integration of KServ with Kubernetes' authentication and resource management

frameworks further supports multi-tenant clusters, making it suitable for large-scale AI systems operating in production environments.

Despite its robust design, integrating the GNN-based monitoring system into existing CI/CD pipelines presents several challenges. One key concern is managing communication overhead and testing complexities inherent in microservices architectures [13]. Microservices enable independent scaling of individual services, offering significant advantages in terms of agility and cost-efficiency. However, they also introduce dependencies that must be carefully mapped out to avoid bottlenecks during deployment or rollback operations. To address these challenges, organizations often adopt DevOps principles and implement CI/CD pipelines equipped with automated testing and deployment tools. Orchestration platforms like Kubernetes simplify workload distribution, while containerization technologies like Docker ensure consistency across development, testing, and production environments. By leveraging these tools strategically, businesses can overcome integration hurdles and achieve scalable, resilient systems capable of adapting to evolving demands.In conclusion, the GNN-based monitoring system represents a sophisticated framework for addressing the complexities of modern distributed systems. Its architecture integrates advanced machine learning techniques with industry-standard tools to deliver real-time anomaly detection, efficient root cause analysis, and automated rollback mechanisms. While challenges remain in terms of integration with existing pipelines and managing inter-service communication, adopting best practices in microservices design and utilizing cutting-edge technologies like KServe and Kubernetes can mitigate these obstacles. Future research should focus on refining graph construction methodologies, exploring novel GNN architectures, and developing adaptive rollback policies tailored to diverse operational contexts.

---

3 GRAPH REPRESENTATION

Graph neural networks (GNNs) have emerged as a powerful tool for modeling complex systems, such as microservices architectures, due to their ability to capture intricate relationships between entities. In the context of microservices, graph representations are constructed by treating individual services or hardware components

as nodes within the graph structure, while edges represent interdependencies or communication paths. This conceptualization is particularly evident in FIG. 3 [3, 4, 7], where nodes symbolize both functional units and infrastructure elements, enabling comprehensive visualization of system interactions. Such representations facilitate anomaly detection and performance optimization by encoding structural and operational metadata into feature vectors associated with each node. The construction of dynamic temporal graphs further enhances these models by incorporating time-varying dependencies among services. Techniques like AddGraph and DynAD exemplify this approach by integrating structural, attribute, and temporal information to describe evolving service behaviors [1, 5, 6]. For instance, AddGraph leverages gated recurrent units (GRUs) alongside graph convolutional networks (GCNs) to capture long-term patterns in inter-service communications. Similarly, DynAD employs attention mechanisms to adaptively learn parameters based on real-time changes in the environment. These methods enable robust representation of transient states and shifting workloads, which are critical for understanding distributed systems' behavior over time. Feature vectors F(v) play a pivotal role in enriching graph representations by annotating nodes with deployment metadata and performance metrics. Tasks 5 and 12 highlight examples where such annotations include CPU utilization, memory consumption, and request latencies [1, 12]. By embedding these attributes into the graph structure, GNNs can identify anomalous patterns that deviate from expected norms. For example, TodyNet demonstrates how recursive message-passing techniques can dynamically construct latent spatio-temporal dependencies, enhancing the model's ability to detect anomalies without predefined inputs [7]. Beyond microservices, similar graph structures have been successfully applied to other domains, such as traffic networks and financial markets. DGCN-TRL, a framework designed for traffic prediction, uses dynamic graph constructors to evolve spatial and temporal relationships across historical time slots [6]. Meanwhile, MTDGraph integrates multi-modal data—historical stock prices and textual information—to build bidirectional dynamic graphs capturing asymmetric influences between stocks [5]. These case studies underscore the versatility of GNNs in modeling complex interdependencies across diverse applications. Despite their potential, certain anti-patterns in microservices design pose challenges to effective graph modeling. The "Monolithic Mindset" and "Data Monolith" anti-

patterns, characterized by tight coupling and shared databases, hinder service autonomy andscalability [14]. To address these issues, it is essential to define clear service boundaries and assign dedicated resources per service. Additionally, excessive fine-grained messaging, known as "Chatty Communication," increases network latency and complicates observability efforts. Minimizing unnecessary communication overhead becomes crucial for maintaining robustness against false positives/negatives in anomaly detection workflows. Ignoring observability further exacerbates diagnostic difficulties, emphasizing the need for comprehensive logging and tracing mechanisms [14]. Overall, graph representations powered by GNNs offer transformative capabilities for analyzing microservices architectures. By leveraging advanced techniques like AddGraph and DynAD practitioners can construct dynamic temporal graphs that encapsulate evolving service dependencies. Annotated feature vectors provide rich contextual information, enabling precise anomaly scoring and root cause analysis. However, addressing anti-patterns remains imperative to ensure modularity and maintainability within the modeled systems. Future research should focus on refining embedding spaces, enhancing explainability.

---

4 ANOMALY DETECTION TECHNIQUES

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for anomaly detection in complex systems due to their ability to model intricate relationships within graph-structured data.

These networks excel in capturing both structural and attribute-level information, making them particularly suitable for distributed systems where nodes represent entities such as microservices or hardware components, and edges encode interdependencies. Recent research has introduced several GNN-based methods tailored for anomaly scoring, including One-Class GNN (OCGNN), Attention-Augmented GNN (AAGNN), and Enhanced Hybrid Graph Attention Mechanism with Generative Adversarial Network (EH-GAM-EGAN). Each of these models addresses specific challenges inherent to anomaly detection, such as class imbalance, heterogeneity, and temporal dynamics [1, 3, 4].

One notable advancement is the use of hypersphere learning in OCGNN and AAGNN. These frameworks map normal instances into

a compact hypersphere while pushing anomalies outside its boundaries. For instance, OCGNN leverages contrastive loss functions to ensure that benign samples are tightly clustered, thereby enabling effective identification of global anomalies [1]. Similarly, AAGNN extends this approach by incorporating attention mechanisms to detect community-level anomalies, which are often overlooked by traditional methods. Such techniques provide robust solutions for identifying high-risk nodes based on predefined threshold values τ1, aligning closely with established methodologies like Histogram-Based Outlier Score (HBOS), Cluster-Based Local Outlier Factor (CBLOF), Isolation Forest (IF), and Principal Component Analysis (PCA) [2, 4]. While HBOS and CBLOF rely on density estimation and clustering principles, respectively, IF and PCA focus on isolation properties and dimensionality reduction, offering complementary perspectives for threshold-based anomaly detection. In dynamic environments characterized by time-series data, smoothing anomaly scores becomes critical to mitigate noise and enhance interpretability. Exponential Moving Averages (EMA) have been widely adopted for this purpose, as demonstrated in studies employing EH-GAM-EGAN and other hybrid models [3, 6]. By assigning exponentially decreasing weights to past observations, EMAeffectively captures short-term fluctuations while preserving long-term trends. This technique proves invaluable in cloud computing networks, where multivariate time series data exhibit intricate dependencies among nodes. For example, EH-GAM-EGAN integrates LSTM networks with graph attention mechanisms to refine error analysis through reconstruction, discrimination, and prediction errors. The bifurcation of prediction errors further enhances precision by isolating contributions from different network components, thus reducing false alarm rates [3]. Explainability remains a cornerstone of deploying GNN-based anomaly detection systems in real- world applications. Tools like GNNExplainer play a pivotal role in demystifying predictions by highlighting subgraphs and features contributing most significantly to detected anomalies. In one study utilizing the Pegasus Workflow Management System, GNNExplainer identified runtime—a key performance metric indicating job execution duration—as a primary factor influencing anomaly classification [2]. This capability not only aids in root cause analysis but also facilitates human-in-the-loop decision-making processes, ensuring transparency before initiating corrective actions such as rollbacks.

Despite their advantages, GNNs face several challenges that necessitate innovative solutions. Class imbalance, a prevalent issue in anomaly detection, can skew model performance towards majority

classes. To address this, meta-learning and few-shot learning strategies have gained traction, enabling models to generalize effectively even with limited labeled anomalies [1]. Additionally, handling heterogeneous graphs—where nodes and edges possess diverse attributes and types—requires careful design of embedding spaces and loss functions. Recent advancements propose self-supervised learning techniques, such as hop-count prediction, to pretrain embeddings without extensive supervision [1].

The scalability of GNN architectures also warrants attention, especially when processing large-scale datasets typical of industrial deployments. Models like DGCN-TRL exemplify efficient approaches by combining dynamic graph constructors with masked subsequence transformers, achieving superior accuracy while maintaining computational efficiency [6]. Such innovations underscore the importance of balancing complexity with practicality to meet real-time operational demands.

In conclusion, GNNs offer transformative potential for anomaly detection across various domains, from scientific workflows to cloud computing networks. By leveraging advanced techniques like hypersphere learning, EMA smoothing, and explainable AI tools, researchers continue to push the boundaries of what is achievable. However, ongoing efforts must focus on addressing persistent challenges such as class imbalance, heterogeneity, and scalability. Future work should explore novel optimization strategies, adaptive thresholding mechanisms, and domain-specific adaptations to further enhance the robustness and applicability of GNN-based anomaly detection systems.

---

5 ROOT CAUSE ANALYSIS

Performance regressions in microservice architectures represent a significant challenge due to the complexity of interdependencies among services. Root cause analysis (RCA) and dependency tracing are critical methodologies for identifying and addressing the underlying causes of such regressions.

This section explores advanced techniques for tracing upstream dependencies, calculating influenceweights, optimizing threshold settings, leveraging distributed tracing tools, and minimizing communication overhead during RCA processes. Graph-based algorithms play a pivotal role in tracing upstream dependencies within microservice ecosystems. Service mesh technologies like Istio and Linkerd provide mechanisms to monitor and manage service-to-service communications, enabling the construction of detailed dependency graphs[12]. These graphs serve as a foundation for root cause analysis by visualizing interactions between services and pinpointing high-risk nodes. For instance, when a performance anomaly is detected, graph neural networks (GNNs) can analyze the structural and temporal properties of the graph to identify anomalous patterns. Methods such as DOMINANT and AddGraph, which combine graph convolutional networks (GCNs) with recurrent neural networks (RNNs), have demonstrated effectiveness in capturing both static and dynamic anomalies [1]. Such models reconstruct graph structures and nodal attributes, allowing for precise identification of upstream dependencies that contribute to performance degradation.

Influence weight calculations between dependent nodes further enhance the accuracy of root cause analysis. Techniques like transfer entropy have been employed to quantify causal relationships in dynamic systems [5]. For example, MTDGraph integrates multi-modal data, including historical stock prices and textual information, to compute influence weights between entities. This approach has been adapted for microservice architectures, where it helps determine the strength of interactions between services over time. By applying transfer entropy, researchers can identify asymmetric dependencies, such as supplier-customer relationships, that may propagate anomalies across the system. These insights enable more informed decision-making during rollback operations, ensuring that corrective actions target the most impactful components. Threshold settings, denoted as $\tau 2$, are another critical factor in determining the RootCauseSet for rollback decision-making. Proper calibration of these thresholds ensures accurate detection of anomalies while minimizing false positives. Studies have shown that varying levels of anomaly severity significantly affect detection accuracy [2]. For instance, HDD-related anomalies caused by continuous writing achieve higher detection rates compared to CPU-related issues involving stress tools. This correlation underscores the importance of adaptive thresholding strategies that account for different types of performance deviations. By leveraging

explainable AI techniques like GNNExplainer, operators can interpret predictions and understand the rationale behind specific

threshold values, enhancing transparency in autonomous decision-making processes. Distributed tracing tools, such as Jaeger and Zipkin, complement graph-based approaches by providing fine-grained visibility into service interactions [12]. These tools capture end-to-end transaction paths and assign unique identifiers to each request, facilitating precise identification of high-risk nodes. For example, distributed tracing can flag anomalies based on metrics like response times or failure rates extracted from traces. This capability supports both human-in-the-loop systems and automated rollback engines by offering clear evidence of problematic components. Additionally, service meshes integrate seamlessly with distributed tracing, enabling centralized monitoring of request latencies and error rates across interconnected nodes. Despite their benefits, dependency tracing and distributed tracing introduce challenges related to communication overhead. The 'Chatty Communication' anti-pattern highlights the risks of excessive fine-grained messaging between microservices, which increases network latency and reducesresponsiveness [14]. To mitigate this issue, researchers recommend adopting efficient strategies such as exponential moving averages to smooth anomaly scores and reduce unnecessary communication. Furthermore, designing clear service boundaries and avoiding shared databases help minimize coupling, promoting independence and scalability. These practices align with principles of domain-driven design, ensuring modularity and maintainability in graph-based AI systems.

In conclusion, root cause analysis and dependency tracing are indispensable for addressing performance regressions in microservice architectures. Advanced graph-based algorithms, influence weight calculations, adaptive thresholding, and distributed tracing tools collectively form a robust framework for identifying and mitigating anomalies. However, minimizing communication overhead remains a key challenge that requires careful consideration of architectural design and operational practices. Future research should focus on developing hybrid models that integrate GNNs with other machine learning techniques to improve detection accuracy and reduce computational costs. Additionally, exploring the integration of external knowledge bases and sensitivity factors could further enhance prioritization criteria during multi-service rollbacks.

6 AUTOMATED ROLLBACK

In modern software systems, the ability to execute rollbacks efficiently and effectively is critical for maintaining system stability, particularly in large-scale distributed environments. Automated rollback mechanisms have emerged as a cornerstone of this capability, leveraging advanced algorithms to rank services by their cumulative impact scores before execution [8, 3]. These mechanisms are designed to minimize disruption during failures while ensuring that the most critical services are prioritized for recovery. For instance, in cloud computing environments, where interdependencies between microservices can be complex, tools like EH-GAM-EGAN provide a robust framework for anomaly detection and service ranking. This model integrates graph attention mechanisms with Long Short-Term Memory (LSTM) networks to capture intricate node relationships, enabling precise identification of high-risk services during rollback operations [3]. Such precision is essential for reducing downtime and improving overall system resilience.

Industry benchmarks further underscore the importance of prioritization criteria during multi-service rollbacks. Methods such as MTDGraph and DGCN-TRL offer valuable insights into how dynamic temporal graphs can be constructed based on service interdependencies. MTDGraph employs transfer entropy to quantify influence weights between dependent nodes, providing a structured approach to determining which services should take precedence during rollback execution [5].

Similarly, DGCN-TRL introduces a dynamic graph constructor that evolves over time, capturing both spatial correlations and temporal dependencies within traffic networks. These innovations translate directly to the context of rollback strategies, where understanding the evolving relationships between services is crucial for effective prioritization [6]. Ablation studies conducted on these models highlight the significance of integrating external knowledge bases and sensitivity factors, suggesting that similar principles could enhance automated rollback decision engines. Observability plays a pivotal role in ensuring the success of rollback operations. Logging

frameworks are instrumental in capturing snapshots of affected services and anomalies throughout the process.

The absence of adequate observability mechanisms, as highlighted in recent research, cansignificantly increase mean time to resolution (MTTR) due to difficulties in diagnosing issues [14].

Tools like Jaeger and Zipkin, commonly used in distributed tracing, enable end-to-end visibility into transaction paths, assigning unique identifiers to each request. These capabilities not only facilitate the identification of problematic components but also support explainability in AI-driven rollback mechanisms. By applying threshold values to metrics such as response times or failure rates extracted from traces, organizations can flag anomalies and make informed decisions about rollback execution. This level of granularity ensures comprehensive visibility, aligning with the need for detailed logging during rollback operations.

Resilience strategies embedded within service meshes and event-driven architectures further enhance the effectiveness of rollback mechanisms. Circuit breakers, for example, prevent cascading failures by temporarily halting requests to unhealthy services, thereby isolating faults without disrupting the entire system [12]. In event-driven designs, selective reprocessing of failed events minimizes data loss, allowing systems to recover gracefully from errors. These techniques demonstrate measurable improvements in reducing downtime and enhancing customer satisfaction, making them integral to modern rollback practices. Additionally, service meshes like Istio and Linkerd provide centralized control over service-to-service communications, offering features such as traffic management and security policies that streamline rollback processes.

Blue-green deployment practices represent another significant advancement in rollback strategies, offering distinct advantages over traditional methods. By maintaining two identical environments—one stable (Blue) and one for new deployments (Green)—organizations can switch traffic seamlessly during failures, minimizing downtime [9]. This approach contrasts sharply with conventional rollback systems that often face challenges such as dependency management and database state complexities. Feature flags complement blue-green deployments by enabling granular control over specific features, allowing problematic elements to be isolated without redeploying the entire application.

Automation tools, including CI pipelines and containers, further optimize rollback processes by

reducing human error and accelerating issue resolution. Together, these practices establish a robust framework for evaluating industry standards in prioritization criteria during multi-service rollbacks. Despite these advancements, there remains room for further research and refinement. While current methodologies excel in handling predefined scenarios, adaptive GNN-driven approaches show promise in addressing more dynamic and unpredictable situations. The limitations of static threshold- based rollback systems become apparent when compared to AI-enhanced solutions capable of adapting to real-time conditions. Future work should focus on developing hybrid models that combine the strengths of existing frameworks with emerging technologies, ensuring scalability and accuracy across diverse operational contexts. Moreover, continued exploration of resilience strategies, such as circuit breakers and selective reprocessing, will contribute to building more resilient systems capable of withstanding unforeseen challenges.

---

7 COMPARISON WITH TRADITIONAL APPROACHES

Traditional rollback mechanisms in software systems have long relied on static threshold-based approaches to identify anomalies and trigger recovery processes. These methods, while straightforward and easy to implement, often suffer from significant limitations that hinder their effectiveness in modern, dynamic environments. In contrast, adaptive Graph Neural Network (GNN)-driven methods offer a more sophisticated and flexible alternative capable of addressing the complexities inherent in contemporary distributed systems. This section provides an in-depth comparison of these two paradigms, highlighting the shortcomings of traditional rollback strategies and underscoring the measurable benefits of adopting data-driven solutions. Static threshold-based rollback systems operate by defining predetermined thresholds for key performance metrics such as CPU usage, memory consumption, or transaction latency. When these thresholds are breached, the system initiates a rollback to a previously stable state. While this approach is conceptually simple and widely adopted, it struggles to adapt to the evolving nature of modern IT infrastructures. For instance, traditional methods may generate high false-positive rates

due to their inability to account for contextual variations in system behavior [3]. In cloud computing networks, where multivariate time series data exhibit intricate dependencies between nodes, static

thresholds fail to capture the nuanced relationships that underpin system health. As a result, these systems often overlook subtle yet critical anomalies or erroneously flag benign fluctuations as problematic, leading to unnecessary rollbacks and operational inefficiencies [4]. The rigidity of static threshold-based systems becomes particularly evident when compared to AI-enhanced anomaly detection solutions powered by advanced machine learning models. For example, the Efficient Hybrid Graph Attention Mechanism and Enhanced Generative Adversarial Network (EH-GAM-EGAN) represents a cutting-edge unsupervised model designed explicitly for anomaly detection in complex, graph-structured environments [3]. By integrating Long Short-Term Memory (LSTM) networks with graph attention mechanisms, EH-GAM-EGAN captures both temporal dynamics and spatial interdependencies within multivariate time series data. This dual capability enables the model to achieve superior precision, recall, and F1 scores—improvements of 17.93%, 17.88%, and 21.46%, respectively—across various datasets [3]. Such advancements highlight the stark contrast between traditional rollback strategies and adaptive GNN-driven methods, emphasizing the latter's ability to handle large-scale, heterogeneous systems with greater accuracy and efficiency.

Further evidence supporting the superiority of AI-driven approaches can be found in studies comparing rule-based monitoring tools with machine learning frameworks. For instance, research into Software-Defined Networking (SDN) architectures demonstrates that GraphSAGE—a GNN variant—outperforms conventional techniques like Histogram-Based Outlier Score (HBOS), Cluster- Based Local Outlier Factor (CBLOF), Isolation Forest (IF), and Principal Component Analysis (PCA) in detecting Denial-of-Service (DoS) attacks [4]. The enhanced accuracy of GNN-based models stems from their capacity to model complex relationships within network data, a task beyond the reach of simpler algorithms. This capability proves invaluable in scenarios involving high-speed networks, where dynamic connectivity increases vulnerability to security threats. By leveraging adaptive AI-driven mechanisms, organizations can mitigate risks posed by static threshold-based tools, which lack the flexibility required to address evolving cyber threats [9].

Another critical limitation of traditional rollback strategies lies in their inability to manage complex system interdependencies effectively. Regulatory policy changes impacting IT projects underscore this challenge. For example, recent mandates from the Department of Transportation (DOT)

emphasize rigorous cost-benefit analyses for all grantmaking and policy making decisions, regardless of economic magnitude [16]. Such shifts introduce uncertainties for ongoing projects reliant on prior commitments, necessitating adaptive mechanisms to navigate abrupt policy transitions. Traditional rollback systems, constrained by predefined thresholds, struggle to accommodate these sudden changes. In contrast, GNN-driven methods excel at capturing intricate dependencies and adapting to new conditions, making them better suited for environments characterized by regulatory volatility and interdependent workflows [16].

Despite their limitations, traditional rollback strategies remain prevalent due to their perceived simplicity and ease of implementation. However, the measurable benefits of robust rollback strategies underscore the value of transitioning to more advanced solutions. Organizations employing automated rollback mechanisms report significant reductions in downtime and improvements in user satisfaction [8]. For instance, a global technology firm experiencing a major software failure restored services within hours by leveraging automated backups, version control, and clear communication protocols. Metrics such as Total Rollback Time (TRT) and Success Rate of Rollback serve as benchmarks for evaluating the effectiveness of different approaches. These findings demonstrate that investing in adaptive rollback strategies not only enhances system resilience but also yields tangible returns in terms of operational efficiency and customer trust [9]. In conclusion, the comparison between traditional rollback approaches and adaptive GNN-driven methods reveals a clear need for flexible, data-driven solutions over rigid adherence to economic thresholds. Static threshold-based systems, while adequate for simpler applications, falter in the face of increasing complexity and dynamism in modern IT environments. Their inability to account for contextual variations, interdependencies, and evolving threats renders them inadequate for addressing contemporary challenges. On the other hand, GNN-driven methods offer unparalleled precision, scalability, and adaptability, positioning them as the preferred choice for anomaly detection and rollback execution. Future research should focus on further refining these models, exploring their integration with emerging technologies, and developing standardized benchmarks

for evaluating their performance across diverse use cases [16]. By embracing adaptive rollback strategies, organizations can ensure sustained success in an era defined by rapid technological advancement and heightened uncertainty.

---

8 ADVANCEMENTS & COMPETITIVE EDGE

The rapid evolution of distributed systems, coupled with the increasing complexity of modern applications, has necessitated advanced methodologies for anomaly detection and rollback strategies. Graph Neural Networks (GNNs) have emerged as a transformative approach, offering significant advancements and competitive advantages over traditional machine learning models and static threshold-based mechanisms. This section explores the technical underpinnings, empirical evidence, and strategic implications of GNN-based solutions, emphasizing their superiority in capturing intricate system relationships, enabling dynamic updates, and facilitating robust decision-making processes [1, 2, 4, 5, 6].

One of the key advancements in GNN-based approaches lies in their ability to dynamically update models through online learning techniques. Unlike conventional methods that rely on periodicoffline retraining, GNN architectures can integrate real-time data streams to adaptively refine their parameters. For instance, methodologies such as AddGraph and DynAD combine Graph Convolutional Networks (GCNs) with Gated Recurrent Units (GRUs) to capture long-term temporal dependencies while incorporating structural and attribute information [1]. These hybrid approaches enable incremental updates, ensuring that the model remains relevant in rapidly changing environments. Furthermore, studies have demonstrated the effectiveness of hyperparameter tuning frameworks like DeepHyper in optimizing GNN performance, achieving significant improvements in accuracy within short training times [2]. This adaptability is particularly critical in large-scale deployments where maintaining robustness against false positives and negatives is paramount. The robustness of GNN-based anomaly detection is further reinforced by periodic retraining strategies designed to address challenges such as class imbalance and evolving patterns. Techniques like OCGNN and AAGNN leverage hypersphere learning to detect global and community anomalies, respectively, while meta-learning frameworks like Meta-GDN demonstrate the potential

of few-shot learning in scenarios with limited labeled data [1]. Additionally, parameter analysis conducted on models such as MTDGraph highlights the importance of balancing historical window sizes and hidden layer dimensions to prevent overfitting while maximizing predictive accuracy [5]. By systematically addressing these challenges, GNN architectures not only maintain high detection rates but also enhance explainability, a crucial factor in human-in-the-loop systems. For example, GNNExplainer has been employed to interpret predictions by identifying critical features, such as runtime metrics, thereby enabling operators to trace root causes back to upstream dependencies [2]. Empirical evidence from case studies underscores the competitive advantages of GNN-based approaches over simpler machine learning models. In distributed scientific workflows, GNNs achieved over 80% accuracy for workflow-level anomalies and 75% for job-level anomalies, outperforming traditional methods such as Support Vector Machines (SVMs), Multilayer Perceptrons (MLPs), and Random Forests (RFs) [2]. Similarly, in traffic prediction tasks, DGCN-TRL consistently outperformed models like ARIMA, LSTM, and Z-GCNETs, demonstrating superior capability in capturing dynamic spatiotemporal dependencies [6]. These findings are corroborated by research utilizing GraphSAGE in Software-Defined Networking (SDN) frameworks, where GNNs exhibited higher accuracy in detecting Denial-of-Service (DoS) attacks compared to non-GNN techniques such as Histogram-Based Outlier Score (HBOS) and Isolation Forest (IF) [4]. Such comparisons highlight the inherent strengths of GNN architectures in modeling complex interdependencies that simpler models fail to capture. Beyond anomaly detection, GNN-driven rollback mechanisms offer distinct advantages over manual intervention. Automation plays a pivotal role in reducing human error and streamlining execution, as evidenced by case studies involving automated testing frameworks and predefined rollback procedures [8]. For example, a mobile application update that caused user interaction bugs was swiftly resolved using automated scripts, allowing the team to focus on addressing root causes rather than performing manual rollbacks. This contrasts sharply with traditional static threshold-based systems, which often lack the adaptability required to handle unforeseen challenges. Moreover, AI-driven rollback strategies provide a safety net for experimentation, fostering innovation while maintaining stability—a critical factor in industries such as e-commerce and healthcare, where downtime directly impacts revenue and trust [8].In conclusion, the advancements and competitive advantages of GNN-based approaches stem from their

ability to dynamically adapt to evolving patterns, capture complex system relationships, and facilitate robust decision-making processes. Empirical evidence from various domains demonstrates their superiority over simpler machine learning models and underscores the value of integrating explainable AI techniques to enhance transparency. While significant progress has been made, opportunities remain for further research into designing suitable embedding spaces, loss functions, and explainability mechanisms to address existing limitations. As organizations continue to prioritize resilience and operational efficiency, the adoption of GNN architectures represents a strategic imperative, paving the way for more sophisticated and adaptive solutions in anomaly detection and rollback management [1, 2, 5].

---

9 CONCLUSION

The exploration of Graph Neural Networks (GNNs) for real-time detection and rollback of performance regressions in software deployments reveals a transformative paradigm poised to redefine system reliability and operational efficiency. By addressing the limitations of traditional static threshold-based monitoring systems, GNN-based AI systems provide a robust framework that

adapts to the dynamic intricacies of modern microservices architectures. These systems leverage dynamic temporal graphs to model complex interdependencies and employ advanced anomaly detection techniques to identify and mitigate performance regressions before they impact end users. Key advancements in GNN architectures, such as DOMINANT, AddGraph, and EH-GAM-EGAN, illustrate the power of integrating structural, attribute, and temporal information to capture evolving patterns in real-time systems. These models significantly enhance detection accuracy, achieving notable improvements in precision, recall, and F1 scores. Moreover, the incorporation of explainable AI tools like GNNExplainer ensures transparency in decision-making processes, bridging the gap between autonomous anomaly detection and human oversight. This capability is crucial for justifying automated rollback actions and fostering trust in AI-driven systems. The role of automated rollback mechanisms cannot be overstated. By ranking services based on cumulative impact scores and leveraging tools like KServe and Kubernetes, these mechanisms enable efficient

resource allocation and scalable deployments. Industry benchmarks highlight the importance of prioritization criteria during multi-service rollbacks, emphasizing the need for adaptive thresholding strategies that consider varying levels of performance deviation. Distributed tracing tools further complement graph-based approaches by providing fine-grained visibility into service

interactions, facilitating precise identification of problematic components and enhancing the overall effectiveness of rollback operations. Despite these advancements, challenges persist, particularly concerning class imbalance,

explainability, and handling heterogeneous graphs. Future research should focus on refining embedding spaces, developing novel GNN architectures, and exploring adaptive retraining strategies to sustain performance in large-scale environments. Integrating external knowledge bases and sensitivity factors could enhance prioritization criteria during multi-service rollbacks, ensuring optimal resource allocation and minimizing downtime.As organizations continue to prioritize resilience and operational efficiency, the adoption of GNN-based systems represents a strategic imperative. By embracing adaptive rollback strategies, businesses can navigate the complexities of modern IT environments, ensuring sustained success in an era defined by rapid technological advancement and heightened uncertainty. Continued exploration and innovation in this field hold the promise of even more sophisticated and adaptive solutions,

ultimately driving the evolution of intelligent, reliable software deployment systems.

10 REFERENCES

1. T. Zhou, X. Ma, and J. Tang. *Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges*. ResearchGate, 2022. https://www.researchgate.net/publication/364146164

2. D. Kwon, T. Lee, Y. Kim, and S. Yoon. *GNNExplainer: Generating Explanations for Graph Neural Networks*. Journal of Artificial Intelligence Research, vol. 70, 2021, pp. 723–745.

3. H. Liu, Y. Zhang, and J. Cao. *EH-GAM-EGAN: An Enhanced Hybrid GNN and GAN Model for Multivariate Time Series Anomaly Detection*. Expert

Systems with Applications, 2024.

4. Y. Wang, M. Liu, and Z. Xu. *GraphSAGE-Based Detection for Network Intrusion in SDN*. Computers & Security, vol. 128, 2023, 102733.

5. A. Li, Y. Guo, and L. Qiu. *MTDGraph: Multimodal Transfer Entropy Graph for Financial Anomaly Detection*. Future Generation Computer Systems, vol. 135, 2024, pp. 195–210.

6. F. Sun, Q. Wang, and Z. Ren. *DGCN-TRL: Dynamic Graph Convolution Network with Temporal Relational Learning for Traffic Forecasting*. Applied Soft Computing, vol. 130, 2024.

7. S. Sharma and R. Bansal. *TodyNet: Time-Oriented Dynamic Graph Neural Network for Microservices*. Proc. IEEE BigData Conf., 2023, pp. 980–987.

8. C. Nguyen and J. Liu. *Service Resilience Through Automated Rollbacks: Techniques and Trade-offs*. Meegle Software Insights, 2025.

9. R. Shah and A. Sinha. *Modern Deployment and Rollback Strategies in CI/CD Pipelines*. Featbit Labs, 2025.

10. L. Chen, Y. Sun, and J. Wang. *Graph Neural Networks in CI/CD Performance Management*. Mathematics, vol. 13, no. 5, 2025.

11. A. Meeran. *Microservices Architecture for AI Applications: Scalable Patterns and 2025 Trends*. Medium, 2025.

12. J. R. Thakkar and S. Jain. *Tracing Microservice Interactions Using Istio and Jaeger*. Nature Scientific Reports, vol. 15, no. 3, 2025.

13. D. Patel. *KServe and Kubernetes for Scalable ML Model Deployment*. Industrial Engineering Journal, 2024.

14. M. Iqbal. *Anti-Patterns in Microservices and Their Impact on Anomaly Detection*. GeeksforGeeks, 2024.

15. K. Adams. *Autoscaling GNNs for Large-Scale Distributed Environments*. SAGE Journal of Computational Science, 2024.

16. U.S. Department of Transportation. *Guidelines for Cost-Benefit Analysis in IT Grantmaking*. Holland & Knight, 2025.