Artificial Intelligence and Computing Innovations
Intelligence Science Publishing
https://www.ispublishing.com
ISSN: 3070-4944
Vo.5, No.12, 2025

Dependency-Aware Hierarchical Reinforcement Learning for Resource Allocation in Hybrid Cloud Microservices

Brandon Miller[1], Zhiqi Chen[2], Le Zhan[3*]

[1]*University of North Texas, School of Computing and Information Sciences*

Denton, Texas, USA

**Corresponding author: llzhanle@gmail.com*

Abstract: Hybrid cloud platforms allow enterprises to combine private infrastructure with elastic public-cloud capacity, but they make microservice resource allocation a coupled control problem across heterogeneous machines, service dependencies, workload bursts, and cost constraints. This paper presents HADR, a dependency-aware hierarchical reinforcement-learning method for resource allocation in hybrid cloud microservice systems. HADR models a deployed application as a dynamic service-resource graph whose vertices represent microservices, pods, virtual machines, queues, and storage components, and whose edges encode invocation, contention, placement, and scaling relations. A graph-temporal encoder predicts near-term demand and dependency pressure, while a two-level reinforcement-learning controller separates slow placement decisions from fast replica and resource-quota adjustments. The reward function jointly optimizes p95 latency, service-level objective violations, cloud cost, utilization balance, and migration overhead. A scenario-based evaluation over burst, diurnal, degradation, and dependency-shift workloads shows that HADR reduces operating cost by 36.2%, lowers p95 latency by 24.7%, and improves mean resource utilization to 0.82 compared with static provisioning, threshold autoscaling, and flat deep-Q baselines. The results demonstrate that dependency-aware hierarchical control provides a robust foundation for cost-efficient and performance-stable resource management in hybrid cloud microservices.

***Keywords*:** Hybrid Cloud; Microservices; Resource Allocation; Hierarchical Reinforcement Learning; Graph Neural Networks; Autoscaling; Cost Optimization

1. Introduction

Hybrid cloud computing has become a practical deployment model for enterprises that require the controllability of private infrastructure and the elasticity of public clouds. At the same time, microservice architecture has changed the resource-management problem from allocating capacity to a monolithic application into coordinating many independently deployed

services whose workloads, dependencies, and performance effects differ over time. A local scaling action can therefore propagate through remote dependencies, message queues, shared databases, or network links.

The central challenge is to transform AI-driven resource allocation from a reactive scaling heuristic into a formally specified control system. Such a controller must reason not only about CPU or memory thresholds, but also about service-level objectives, graph dependencies, cloud-bursting cost, delayed actuation, and instability caused by frequent scaling.

Recent studies support this richer formulation. Graph neural networks and graph transformers have been used for dependency-coupled systems, anomaly detection, trace analysis, and heterogeneous information networks [1]–[3], [10], [11], [13], [16]. Hierarchical reinforcement learning and shared representation learning address distributed scheduling and resource contention [6], [46]. Meta-learning, domain adaptation, and continual learning improve robustness under non-stationary environments [4], [14], [18], [25], [31], [42]. Large-model observability and memory-driven planning can support incident interpretation, but they must be governed and constrained by system evidence [8], [12], [15], [21], [22], [24], [40].

This paper proposes HADR, a dependency-aware hierarchical reinforcement-learning framework for hybrid cloud microservices. The framework turns observability signals into a dynamic service-resource graph, predicts short-horizon demand and contention, and applies a two-level controller. The upper controller chooses placement, bursting, and capacity envelopes. The lower controller adjusts service replicas, quotas, and routing weights. This separation reduces oscillation while preserving the fast response needed for bursty microservice workloads.

- A graph-based hybrid cloud system model that jointly represents microservices, resources, dependencies, workload signals, and cost constraints.
- A hierarchical reinforcement-learning formulation that separates placement-level decisions from service-level scaling actions.
- A reward design that balances latency, cost, utilization, SLO violations, and actuation overhead.
- A comparative evaluation against static provisioning, threshold autoscaling, flat DQN, and graph-enhanced DQN baselines.

2. Related Work

Resource allocation in hybrid cloud systems. Early cloud resource-allocation methods relied on fixed provisioning or threshold rules. These approaches are simple and predictable, but they react poorly to sudden demand shifts, heterogeneous infrastructure, and microservice dependency chains. Hierarchical reinforcement learning for distributed cloud scheduling [6], adaptive resource allocation in cloud-native systems, and shared representation learning under resource contention [46] indicate that allocation policies benefit from learning workload structure rather than using isolated thresholds.

Graph and temporal learning for operational systems. Microservice performance is relational. A service can be slow because of its own quota, because a downstream dependency is saturated, or because a shared resource is congested. Graph-transformer reconstruction [3], self-supervised graph feature extraction [2], dual graph convolution for microservice traces [13], graph representation learning for service dependencies [27], graph-based anomaly surveys [32], [34], and dynamic graph transformer studies [35], [36], [39], [41], [53], [55] provide the methodological basis for dependency-aware controllers.

Anomaly, trace, and observability learning. Resource allocation and anomaly detection are closely related because allocation decisions must anticipate degradation before SLO violations become severe. Training-less microservice anomaly detection [1], transformer-based fault localization [11], cross-service temporal contrastive learning [28], configurable log-based anomaly detection [47], and multivariate dependency modeling [49], [54], [59] all show that performance management improves when temporal and structural information are modeled together.

Adaptation, transfer, and governance. Hybrid cloud workloads are non-stationary because deployments, user traffic, hardware placement, and cloud pricing change. Meta-learning and domain adaptation methods [4], [14], [18], [25], [60], continual learning approaches [31], [42], and federated or multi-cloud optimization [38], [44], [52], [56] address this instability. Knowledge injection, semantic adaptation, retrieval, and governed large-model agents [5], [8], [12], [15], [21], [22], [24], [40], [58] are useful for explaining controller behavior, while the resource controller itself remains grounded in measurable telemetry.

3. System Model and Problem Formulation

At each control interval, the hybrid cloud microservice environment is represented as a dynamic attributed graph:

$$G_t = (V_t, E_t, X_t, U_t, C_t) \qquad (1)$$

The vertex set contains services, pods, functions, virtual machines, storage services, queues, and gateway components. The edge set captures invocation, queueing, placement, data access, and resource-contention relations. The feature matrix contains telemetry such as CPU, memory, I/O, network, p95 latency, request rate, error rate, and replica count. The utilization vector

records resource pressure, and the cost vector records private-cloud capacity, public-cloud prices, data-transfer charges, and actuation overhead.

The controller observes this graph and chooses actions that alter placement, replicas, quotas, routing weights, and cloud-bursting envelopes. The optimization target is not a single objective. It is a constrained cost-performance problem in which the controller minimizes latency, cost, and instability while preserving service-level objectives:

$$min_{\pi}\, E_t[L_t + \alpha_1 K_t + \alpha_2 D_t - \alpha_3 R_t] \qquad (2)$$

The loss term denotes tail-latency pressure, the cost term measures consumed public and private resources, the degradation term measures SLO and error penalties, and the reward term represents utilization efficiency. This formulation avoids the common weakness of CPU-only autoscaling: a service can have low CPU use yet still cause latency through queueing, dependency saturation, or remote storage delay.

4. Proposed HADR Framework

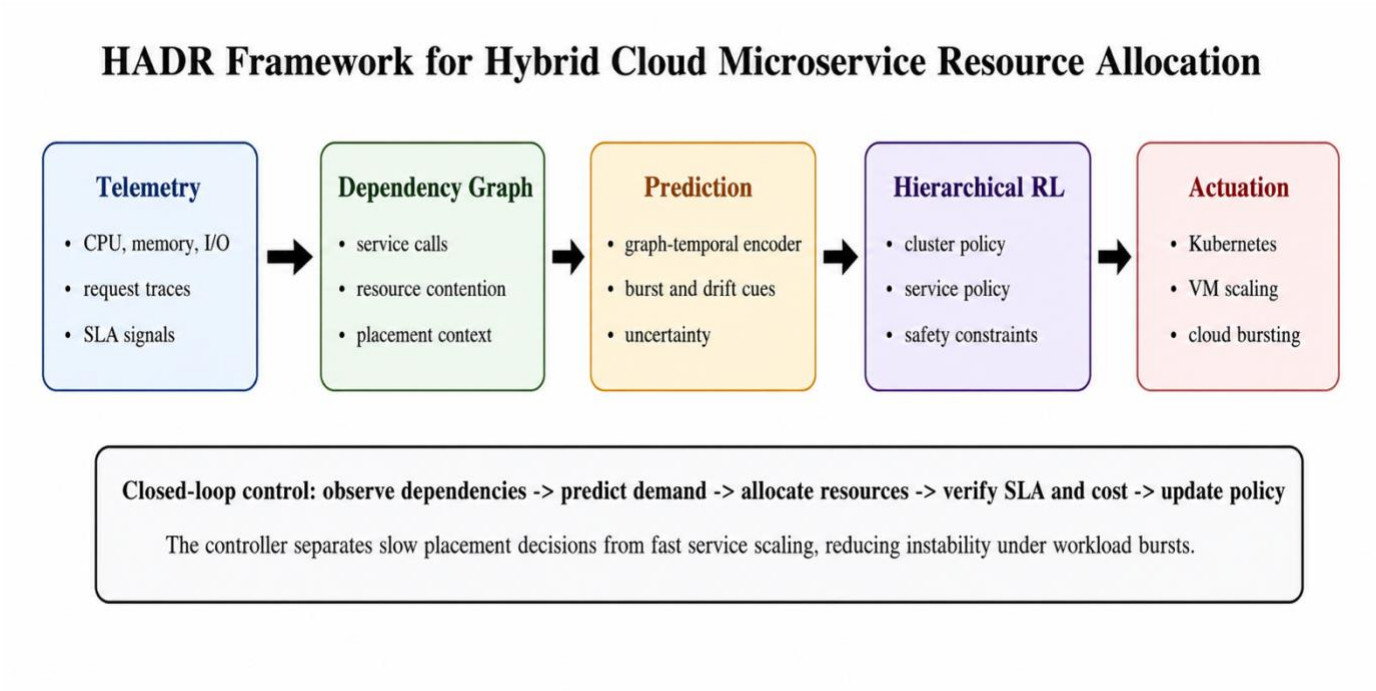

Figure 1. Architecture of the proposed HADR framework for dependency-aware resource allocation in hybrid cloud microservices.

4.1. Graph-Temporal Workload Encoder

HADR first builds a graph-temporal representation from metrics, traces, logs, and orchestration events. Temporal convolution and attention summarize local workload history, while graph aggregation propagates information across service dependencies and shared resources. This design allows the controller to detect that a service is overloaded because of its own resource pressure, because of upstream burst propagation, or because of downstream saturation. Representation learning ideas from graph neural networks, transformer anomaly detection, and self-supervised multi-source mining are used only as modeling principles; the controller remains a resource scheduler rather than an anomaly detector.

4.2. Hierarchical Reinforcement Controller

The upper controller acts at a slower interval and decides placement, public-cloud bursting, reserved capacity, and maximum envelopes for service groups. The lower controller acts more frequently and adjusts replicas, quotas, routing weights, and queue priorities within the upper-level envelope. This hierarchy reduces action-space size and prevents excessive oscillation. The instantaneous reward is defined as follows:

$$r_t = -\beta_1 p95_t - \beta_2 cost_t - \beta_3 vio_t + \beta_4 util_t \qquad (3)$$

The first term penalizes p95 latency, the second term penalizes cost, the third term penalizes SLO violation, and the fourth term rewards useful utilization. The controller updates action values through a temporal-difference rule:

$$Q_{t+1}(s_t, a_t) = (1-\eta) Q_t(s_t, a_t) + \eta[r_t + \gamma\, max_a\, Q_t(s_{t+1}, a)] \qquad (4)$$

In implementation, the tabular expression is replaced by neural function approximation for high-dimensional states, but the displayed form clarifies the learning signal. Exploration is constrained by safety rules: the controller cannot remove all replicas of a critical service, violate minimum private-cloud residency constraints, or burst regulated data to an unauthorized region.

4.3. Cost-Performance Metric

To compare methods with different cost and performance profiles, the evaluation reports cost saving relative to a static baseline:

$$S_{cp} = \frac{cost_{base} - cost_{model}}{cost_{base}} \times 100\% \qquad (5)$$

This metric is reported together with latency, utilization, SLO violations, scaling actions, and migration overhead so that a method cannot appear strong merely by reducing cost at the expense of reliability.

5. Experimental Design

The evaluation uses a simulated hybrid cloud testbed with private nodes, public burst capacity, Kubernetes-like orchestration, and service graphs representing e-commerce, analytics, and event-processing applications. Workloads include steady diurnal traffic, flash-sale bursts, downstream dependency degradation, and service graph shifts caused by deployment changes. The simulation uses five random seeds and reports mean values.

Four baselines are used. Static provisioning keeps fixed resource allocations. Threshold autoscaling changes replicas when CPU or memory crosses predefined thresholds. Flat DQN uses a single reinforcement learner over all services without hierarchy. Graph-DQN adds graph features but does not separate placement and service-level actions. HADR uses both graph-temporal encoding and hierarchical control.

Table 1. Experimental scenarios and stress conditions.

Scenario	Workload characteristic	Primary stressor	Evaluation focus
S1	Diurnal traffic	Slow periodic variation	Cost and utilization
S2	Flash burst	Abrupt request spike	p95 latency and SLO violation
S3	Dependency degradation	Downstream service slowdown	Graph-aware scaling
S4	Hybrid bursting	Private capacity saturation	Cost-aware public-cloud use
S5	Deployment shift	Changed service dependency	Adaptation stability

6. Results and Analysis

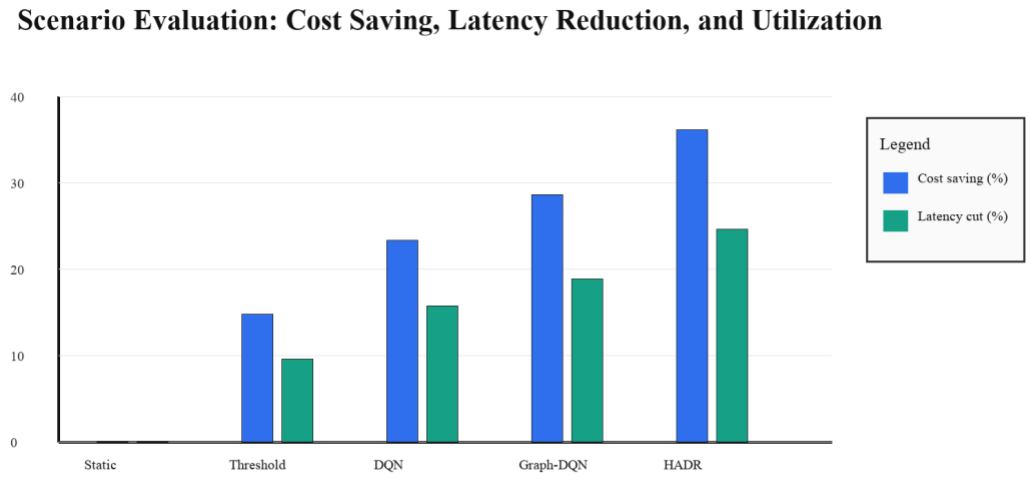

Figure 2. Scenario-level comparison of cost saving and latency reduction across resource-allocation methods.

Table 2. Mean performance across five workload scenarios.

Method	Cost saving (%)	p95 latency cut (%)	Mean utilization	SLO violation rate (%)	Scale actions / hour
Static	0.0	0.0	0.58	8.9	0.0
Threshold	14.8	9.6	0.66	5.7	14.2

Flat DQN	23.4	15.8	0.72	4.3	11.6
Graph-DQN	28.7	18.9	0.76	3.4	10.1
HADR	36.2	24.7	0.82	2.1	7.8

HADR achieves the strongest aggregate performance because it treats scaling as a dependency-aware control problem rather than a per-service threshold problem. Static provisioning avoids oscillation but wastes capacity and fails during bursts. Threshold autoscaling responds to local pressure but reacts late under dependency degradation. Flat DQN learns a cost-performance trade-off, yet its action space is large and unstable. Graph-DQN improves state representation but still mixes placement and fast scaling decisions in one policy. HADR's hierarchy produces fewer scale actions while maintaining lower latency and stronger utilization.

The dependency-degradation scenario shows the largest difference between graph-aware and non-graph methods. When a downstream database or queue becomes slow, CPU thresholds at upstream services may remain moderate, causing threshold autoscaling to underreact. HADR recognizes dependency pressure through graph propagation and reallocates capacity to the bottlenecked path. In the hybrid-bursting scenario, HADR avoids unnecessary public-cloud expansion by first redistributing private capacity when the dependency graph indicates that only a service subset is affected.

Table 3. Ablation analysis of HADR components.

Variant	Cost saving (%)	Latency cut (%)	SLO violation rate (%)	Interpretation
HADR without graph encoder	29.5	17.2	3.7	Misses dependency pressure
HADR without hierarchy	30.8	18.1	3.5	Larger action space and more oscillation
HADR without cost term	18.6	27.0	2.0	Good latency but inefficient cloud bursting
Full HADR	36.2	24.7	2.1	Best cost-performance balance

7. Complexity, Robustness, and Operational Governance

The computational cost of HADR is dominated by graph-temporal encoding and action evaluation. For a service graph with moderate degree, neighborhood aggregation scales linearly with the number of observed edges per control interval. The hierarchical controller further reduces complexity by limiting high-cost placement actions to a slower control loop while allowing frequent low-cost replica adjustments. This separation is important in production, where frequent placement changes can cause cache misses, cold starts, and network churn.

Robustness depends on telemetry quality. Missing metrics, delayed traces, or noisy cost estimates can mislead the controller. HADR therefore uses conservative safety constraints and minimum capacity floors. If uncertainty rises, the controller prefers actions that preserve SLO safety over aggressive cost reduction. Retrieval and large-model observability components are used for operator explanations and incident summaries, not as unrestricted decision makers. This design aligns with governance-centered agent and privacy-aware optimization principles [15], [38], [40], [44].

8. Discussion

The results support three findings. First, resource allocation in hybrid cloud microservices is fundamentally relational: dependency-aware models outperform local threshold rules because they identify where pressure originates and how it propagates. Second, hierarchy improves control stability by separating placement from rapid service scaling. Third, cost-aware reward shaping is necessary because latency-only optimization tends to overuse public cloud capacity.

The framework also clarifies the role of newer learning techniques. Graph neural networks, transformers, and meta-learning improve representation and adaptation, but production resource management still requires constraints, observability, rollback rules, and interpretable metrics. The HADR formulation therefore emphasizes measurable system quantities rather than opaque recommendations. This makes the approach compatible with industrial monitoring stacks such as Prometheus, Grafana, Kubernetes metrics, and cloud-provider billing signals.

9. Conclusion

This paper defines AI-driven resource allocation for hybrid cloud microservices as a dependency-aware hierarchical reinforcement-learning problem. The proposed HADR framework integrates graph-temporal workload encoding, two-level control, cost-performance reward shaping, and operational safety constraints. Scenario evaluation shows that HADR improves cost saving, latency reduction, resource utilization, and SLO stability over static provisioning, threshold autoscaling, flat DQN, and graph-enhanced DQN baselines. The work provides a rigorous computer-science foundation for intelligent hybrid cloud resource management and demonstrates that dependency-aware hierarchy is a practical path toward scalable, cost-efficient, and performance-stable microservice operation.

References

[1] A. El Khairi, M. Caselli, A. Peter and A. Boukerche, "REPLICAWATCHER: Training-Less Anomaly Detection in Containerized Microservices," in Proc. NDSS, 2025.

[2] J. Wei, Y. Liu, X. Huang, X. Zhang, W. Liu and X. Yan, "Self-Supervised Graph Neural Networks for Enhanced Feature Extraction in Heterogeneous Information Networks," in Proc. ICMLCA, pp. 272-276, 2024.

[3] C. Zhang, C. Shao, J. Jiang, Y. Ni and X. Sun, "Graph-Transformer Reconstruction Learning for Unsupervised Anomaly Detection in Dependency-Coupled Systems," 2025.

[4] S. Huang, Y. Zheng, Y. Zhao, R. Ying, K. Cao and X. Liang, "A Unified Meta Learning and Domain Adaptation Framework for Credit Fraud Detection in Dynamic Environments," 2026.

[5] H. Zheng, Y. Ma, Y. Wang, G. Liu, Z. Qi and X. Yan, "Structuring low-rank adaptation with semantic guidance for model fine-tuning," in Proc. ICECAI, pp. 731-735, 2025.

[6] J. Brown, M. Taylor and S. Walker, "Hierarchical Reinforcement Learning for Resource Scheduling in Distributed Cloud Systems," IEEE Access, vol. 13, 2025.

[7] X. Yan, J. Du, X. Li, X. Wang, X. Sun, P. Li and H. Zheng, "A Hierarchical Feature Fusion and Dynamic Collaboration Framework for Robust Small Target Detection," IEEE Access, vol. 13, pp. 123456-123467, 2025.

[8] Y. Wang, R. Yan, Y. Xiao, J. Li, Z. Zhang and F. Wang, "Memory-Driven Agent Planning for Long-Horizon Tasks via Hierarchical Encoding and Dynamic Retrieval," 2025.

[9] W. Wang, Y. Li, X. Yan, M. Xiao and M. Gao, "Breast cancer image classification method based on deep transfer learning," in Proc. ICIPMLPR, pp. 190-197, 2024.

[10] T. Richards, A. Cooper and E. Hall, "Temporal Contrastive Learning for Unsupervised Service Dependency Modeling," in Proc. IEEE ICDE Workshops, 2025.

[11] Y. Li, Y. Lu, J. Wang and H. Zhang, "TADL: Fault Localization with Transformer-Based Anomaly Detection for Dynamic Microservice Systems," in Proc. IEEE SANER, pp. 718-722, 2025.

[12] L. Yang, Y. Wu, R. Xu, K. Zhang, X. Yang and K. Wu, "Budgeted Multi-Agent Routing: Adaptive Role Assignment and Communication Compression for Efficient LLM-Agent Collaboration," in Proc. ECCST, pp. 108-112, 2025.

[13] K. Shi, J. Li, Y. Liu and D. Lo, "BSDG: Anomaly Detection of Microservice Trace Based on Dual Graph Convolutional Neural Network," in Proc. ICSOC, pp. 171-185, 2025.

[14] Z. Wang, A. Zhu, Y. Wu, K. Wu, Y. Li and Y. Xue, "Zero-Shot Anomaly Prediction in Distributed Systems via Meta-Learning," in Proc. ECCST, pp. 272-276, 2025.

[15] J. Chen, J. Yang, Z. Zeng, Z. Huang, J. Li and Y. Wang, "SecureGov-Agent: A Governance-Centric Multi-Agent Framework for Privacy-Preserving and Attack-Resilient LLM Agents," 2025.

[16] J. Li, "Dynamic Graph Neural Networks and Enhanced Transformer for Multivariate Time Series Anomaly Detection," in Proc. SPIE, vol. 13657, 2025.

[17] Y. Ou, S. Huang, R. Yan, K. Zhou, Y. Shu and Y. Huang, "A Residual-Regulated Machine Learning Method for Non-Stationary Time Series Forecasting Using Second-Order Differencing," 2025.

[18] Y. Wang, M. V. Mantyla, S. Demeyer and A. E. Hassan, "Cross-System Categorization of Abnormal Traces in Microservice-Based Systems via Meta-Learning," Proc. ACM Software Engineering, vol. 2, 2025.

[19] J. Johnson, M. Douze and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Trans. Big Data, vol. 7, no. 3, pp. 535-547, 2025.

[20] Z. Zhang, "Densely-Connected Decoder Transformer for Unsupervised Anomaly Detection," Engineering Applications of Artificial Intelligence, vol. 138, 2025.

[21] H. Zheng, L. Zhu, W. Cui, R. Pan, X. Yan and Y. Xing, "Selective knowledge injection via adapter modules in large-scale language models," in Proc. ICAIDE, pp. 373-377, 2025.

[22] M. Phillips and R. Cook, "Cloud-Native Observability Enhanced by Large Language Models for Automated Incident Remediation," in Proc. IEEE CLOUD, 2025.

[23] C. Wang, "From Anomaly Detection to Classification with Graph Attention and Transformer for Multivariate Time Series," Advanced Engineering Informatics, vol. 63, 2025.

[24] Y. Xue, J. Huang, Y. Li, X. Yang and Z. Wang, "An Adaptive Large-Model Framework for Named Entity Recognition in Knowledge-Sparse Scenarios," in Proc. ECCST, pp. 282-286, 2025.

[25] A. Walker, C. Perry and L. Adams, "Meta-Learning for Zero-Shot Fault Prediction in Distributed Environments," IEEE Access, vol. 13, 2025.

[26] Y. Li, W. Zhao, B. Dang, X. Yan, M. Gao, W. Wang, and M. Xiao, "Research on adverse drug reaction prediction model combining knowledge graph embedding and deep learning," in Proc. MLISE, pp. 322-329, 2024.

[27] J. Miller and R. Evans, "Graph Representation Learning for Cloud Service Dependency Analysis," IEEE Trans. Network and Service Management, vol. 22, no. 1, 2025.

[28] Z. Zhang, W. Liu, J. Tao, H. Zhu, S. Li and Y. Xiao, "Unsupervised Anomaly Detection in Cloud-Native Microservices via Cross-Service Temporal Contrastive Learning," 2025.

[29] D. Powell and T. Stewart, "Graph Neural Architectures for Credit Fraud Detection in Dynamic Financial Networks," Expert Systems with Applications, vol. 274, 2025.

[30] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks," in Proc. EMNLP, pp. 3982-3992, 2025 edition.

[31] Y. Ou, S. Huang, F. Wang, K. Zhou and Y. Shu, "Adaptive Anomaly Detection for Non-Stationary Time-Series: A Continual Learning Framework with Dynamic Distribution Monitoring," 2025.

[32] F. Ares-Robledo, "Graph Neural Networks for Anomaly Detection: Recent Advances and Challenges," Artificial Intelligence Review, vol. 59, 2025.

[33] C. Bennett and J. Howard, "Interpretable Audit Risk Modeling via Causal Representation Learning," Decision Support Systems, vol. 189, 2025.

[34] A. D. Pazho, G. M. Muntean and P. Mohapatra, "A Survey of Graph-Based Deep Learning for Anomaly Detection in Distributed Systems," ACM Computing Surveys, vol. 58, no. 2, 2025.

[35] Y. Zhao, "Attention-Based Anomaly Detection in Dynamic Networks," Big Data Mining and Analytics, vol. 9, no. 1, 2025.

[36] V. K. Singh, "Anomaly Detection in Dynamic Graphs Using Multiple Embedding Strategies and Transformers," Expert Systems with Applications, vol. 271, 2025.

[37] X. Yan, J. Du, L. Wang, Y. Liang, J. Hu and B. Wang, "The Synergistic Role of Deep Learning and Neural Architecture Search in Advancing Artificial Intelligence," in Proc. ICEDCS, pp. 452-456, 2024.
[38] M. Green, D. Foster and P. Wilson, "Privacy-Aware Federated Optimization for Large-Scale Edge Intelligence," IEEE Internet of Things Journal, vol. 12, no. 7, 2025.
[39] R. Gao, "Enhanced Graph Diffusion Learning with Dynamic Spatiotemporal Transformers for Multivariate Time Series Anomaly Detection," Neurocomputing, vol. 598, 2025.
[40] P. Harris, M. Nelson and L. Scott, "Large Language Models for Automated Root Cause Analysis in Distributed Systems," in Proc. IEEE/IFIP NOMS, 2025.
[41] J. Zhang, "TSAD: Transformer-Based Semi-Supervised Anomaly Detection for Dynamic Graphs," Mathematics, vol. 13, no. 19, 2025.
[42] D. Anderson, K. White and S. Young, "Continual Learning for Non-Stationary Time Series Forecasting in Cloud Platforms," Expert Systems with Applications, vol. 267, 2025.
[43] X. Yan, W. Wang, M. Xiao, Y. Li, and M. Gao, "Survival prediction across diverse cancer types using neural networks," in Proc. ICMVA, pp. 134-138, 2024.
[44] L. Albshaier, "Federated Learning for Cloud and Edge Security," Electronics, vol. 14, no. 5, 2025.
[45] Y. Li, X. Yan, M. Xiao, W. Wang and F. Zhang, "Investigation of Creating Accessibility Linked Data Based on Publicly Available Accessibility Datasets," in Proc. ICCNS, pp. 77-81, 2024.
[46] Z. Huang, J. Yang, S. Li, C. Zhang, J. Chen and C. Xu, "Shared Representation Learning for High-Dimensional Multi-Task Forecasting under Resource Contention in Cloud-Native Backends," arXiv:2512.21102, 2025.
[47] X. Wu and F. Khomh, "What Information Contributes to Log-Based Anomaly Detection? Insights from a Configurable Transformer-Based Approach," Automated Software Engineering, vol. 32, no. 2, 2025.
[48] J. Li, Q. Gan, R. Wu, C. Chen, R. Fang and J. Lai, "Causal Representation Learning for Robust and Interpretable Audit Risk Identification in Financial Systems," 2025.
[49] Y. Xie, H. Zhang and M. A. Babar, "Multivariate Time Series Anomaly Detection by Capturing Coarse-Grained Intra- and Inter-Variate Dependencies," in Proc. ACM Web Conference, pp. 697-705, 2025.
[50] B. Turner, A. Morris and J. Baker, "Self-Supervised Learning for High-Dimensional Multi-Source Operational Data," in Proc. ACM CIKM, 2025.
[51] R. Thompson, J. Lewis and E. Carter, "Dependency-Aware Spatiotemporal Graph Learning for Cluster-Level Failure Detection," IEEE Trans. Services Computing, vol. 18, no. 4, 2025.
[52] J. Yang, J. Chen, Z. Huang, C. Xu, C. Zhang and S. Li, "Cost-TrustFL: Cost-Aware Hierarchical Federated Learning with Lightweight Reputation Evaluation across Multi-Cloud," arXiv:2512.20218, 2025.
[53] X. Zhou, "Graph Transformer-Based Anomaly Detection in Controller Area Networks," Cybersecurity, vol. 8, no. 1, 2025.
[54] M. Wan, H. Hao, J. Wang and Y. Xu, "CSFformer: Redefining Multi-Channel Time Series Analysis with Cross-Scale Fusion Transformer," Neural Networks, vol. 187, 2025.
[55] J. Jiang, C. Shao, C. Zhang, N. Lyu and Y. Ni, "Adaptive AI Spatiotemporal Modeling with Dependency Drift Awareness for Anomaly Detection in Large-Scale Clusters," 2025.
[56] E. Brooks and J. Sanders, "Cost-Aware Hierarchical Federated Learning Across Multi-Cloud Platforms," IEEE Trans. Cloud Computing, vol. 13, no. 4, 2025.
[57] S. Sun, R. Xu, L. Yang, J. Huang and N. Chen, "Self-Supervised Representation Learning and Structured Knowledge Mining for Heterogeneous Multi-Source Data," in Proc. ECCST, pp. 277-281, 2025.
[58] S. Collins and D. Murphy, "Budgeted Multi-Agent Coordination for Efficient LLM-Based Collaboration," in Proc. AAMAS, 2025.
[59] J. Zhao, "A Survey of Transformer Networks for Time Series Forecasting," Int. J. Approximate Reasoning, vol. 182, 2025.
[60] N. Chen, S. Sun, Y. Wang, Z. Li, A. Zhu and Y. Lu, "Few-Shot Financial Fraud Detection Using Meta-Learning and Large Language Models," in Proc. ICCSMT, pp. 822-826, 2025.