ISSN: 3079-6342

Dynamic Workflow Partitioning via Reinforcement Learning for Edge-Cloud Heterogeneous Systems

Yuexin Zhang*[1] and Caleb Morris[1]

[1] Department of Computer Science, Colorado State University, USA

* Corresponding author: yx.zhang.research@163.com

Abstract

The proliferation of Internet of Things devices and edge computing infrastructure has created unprecedented opportunities for distributed workflow execution across heterogeneous edge-cloud environments. However, optimal workflow partitioning in such dynamic systems remains a significant challenge due to the complexity of resource heterogeneity, network variability, and diverse application requirements. This paper proposes a novel Dynamic Workflow Partitioning framework leveraging Deep Reinforcement Learning to intelligently distribute workflow tasks between edge nodes and cloud data centers. The framework employs a Deep Q-Network architecture enhanced with a Graph Neural Network encoder to capture workflow dependencies and system state representations. Through comprehensive evaluation using real-world workflow applications including CyberShake, Epigenomics, Inspiral, Montage, and Sipht, our approach demonstrates superior performance in minimizing execution time, reducing network overhead, and maintaining quality of service guarantees compared to traditional heuristic-based methods. The experimental results show that the proposed approach achieves up to 32% reduction in average workflow completion time and 41% improvement in resource utilization efficiency across various heterogeneous edge-cloud configurations.

Keywords

Edge computing, cloud computing, workflow partitioning, reinforcement learning, deep Q-network, heterogeneous systems, task scheduling, resource optimization

Introduction

The convergence of edge computing and cloud computing has fundamentally transformed the landscape of distributed computing architectures, creating a continuum of computational resources spanning from resource-constrained edge devices to virtually unlimited cloud data centers [1]. This heterogeneous computing paradigm offers unprecedented opportunities for executing complex workflow applications while addressing critical challenges related to latency, bandwidth consumption, and data privacy [2]. As the volume of data generated by IoT devices continues to grow exponentially, traditional cloud-centric approaches have become increasingly insufficient for applications requiring real-time processing and low-latency responses [3]. Edge computing has emerged as a complementary paradigm that brings computation and storage resources closer to data sources, enabling timely data processing and reducing the burden on network infrastructure [4].

Workflow applications, characterized by directed acyclic graphs of interdependent tasks, represent a fundamental computational paradigm in various domains including scientific computing, business process management, and cyber-physical systems [5]. Scientific workflows such as CyberShake for earthquake hazard characterization, Montage for astronomical image processing, and Epigenomics for genome sequencing exhibit diverse computational patterns and data dependencies that pose unique challenges for distributed execution [6]. The partitioning of workflow tasks across edge-cloud heterogeneous environments presents complexities that distinguish it from traditional scheduling problems. The dynamic nature of edge computing environments, characterized by resource fluctuations, network variability, and diverse device capabilities, necessitates intelligent decision-making mechanisms that can adapt to changing conditions in real time [7].

Reinforcement Learning has emerged as a promising approach for addressing complex decision-making problems in dynamic environments, demonstrating remarkable success in various domains including robotics, game playing, and autonomous systems [8]. The ability of RL agents to learn optimal policies through trial-and-error interactions with their environment makes them particularly suitable for workflow partitioning in edge-cloud systems, where the optimal partitioning strategy depends on multiple time-varying factors including network conditions, resource availability, task dependencies, and application requirements [9]. Deep Reinforcement Learning, which combines the decision-making capabilities of RL with the representational power of deep neural networks, has shown exceptional performance in handling high-dimensional state spaces and complex action spaces that characterize real-world computing systems [10].

Despite the potential advantages of applying DRL to workflow partitioning, several fundamental challenges remain unaddressed in existing research. First, the curse of dimensionality poses significant obstacles when the state space encompasses multiple edge nodes, diverse task types, and dynamic network conditions. Second, the need to capture complex dependencies among workflow tasks while maintaining computational efficiency requires sophisticated state representation mechanisms. Third, ensuring quality of service guarantees while optimizing multiple objectives such as execution time, energy consumption, and cost presents a multi-objective optimization challenge that traditional RL approaches struggle to address effectively. Finally, the deployment of learned policies in real-world edge-cloud environments requires careful consideration of safety constraints, resource limitations, and privacy requirements.

This paper addresses these challenges by proposing a novel Dynamic Workflow Partitioning framework that leverages advanced DRL techniques specifically designed for edge-cloud heterogeneous systems. Our approach integrates Graph Neural Networks with Deep Q-Networks to capture workflow structure and learn effective partitioning policies that adapt to dynamic system conditions. The framework incorporates multi-objective optimization capabilities to balance competing performance metrics while ensuring quality of service constraints are satisfied. Furthermore, we develop specialized techniques for handling the unique characteristics of edge-cloud environments, including resource heterogeneity, network variability, and distributed execution models. The proposed framework has been extensively evaluated using five representative scientific workflow applications with distinct structural characteristics, demonstrating significant improvements over state-of-the-art baseline methods across diverse workflow types and system configurations.

2. Literature Review

The literature on workflow scheduling and partitioning in distributed computing environments has evolved significantly over the past decade, driven by the emergence of cloud computing, edge computing, and the increasing complexity of workflow applications. Early research in this domain focused primarily on cloud-based workflow scheduling, where the emphasis was on optimizing makespan and cost while leveraging the elastic nature of cloud resources [11]. These approaches typically employed heuristic algorithms such as genetic algorithms, particle swarm optimization, and ant colony optimization to find near-optimal solutions to the workflow scheduling problem [12]. While these methods demonstrated effectiveness in cloud environments, they were not designed to handle the unique challenges posed by edge-cloud heterogeneous systems, particularly the dynamic nature of resource availability and network conditions at the edge.

The advent of edge computing has prompted researchers to extend traditional workflow scheduling approaches to accommodate the edge-cloud continuum [13]. Several studies have investigated task offloading strategies that determine whether individual tasks should be executed on edge devices, edge servers, or cloud data centers [14]. These approaches typically formulate the offloading decision as an optimization problem considering factors such as execution time, energy consumption, and network latency [15]. However, most existing offloading strategies operate at the granularity of individual tasks and do not explicitly consider the dependencies and data flow patterns inherent in workflow applications. Furthermore, these methods often rely on accurate predictions of task execution times and network conditions, which are difficult to obtain in dynamic edge-cloud environments [16].

Recent research has begun to explore the application of machine learning techniques to workflow scheduling in edge-cloud systems, recognizing the limitations of traditional heuristic approaches in handling dynamic and uncertain environments [17]. Supervised learning methods have been proposed to predict optimal scheduling decisions based on historical execution data, showing promising results in specific application domains [18]. However, these approaches require extensive labeled training data and may not generalize well to unseen scenarios or changing system conditions [19]. Transfer learning techniques have been investigated to address the generalization challenge, but they still struggle with the continuously evolving nature of edge-cloud environments.

The emergence of Deep Reinforcement Learning has opened new avenues for addressing workflow scheduling challenges in dynamic distributed systems. Several researchers have successfully applied DRL techniques to related problems such as virtual machine placement, container orchestration, and resource allocation in cloud environments [20]. These studies have demonstrated that DRL agents can learn effective policies that adapt to changing conditions and optimize multiple objectives simultaneously [21]. The Deep Q-Network algorithm has become a foundational approach in DRL research, combining Q-learning with deep neural networks to handle high-dimensional state spaces [22]. Variants of DQN, including Double DQN, Dueling DQN, and Rainbow DQN, have been developed to improve sample efficiency and learning stability [23-27].

In the context of edge-cloud workflow scheduling, a growing body of research has begun to explore DRL-based approaches. Researchers have proposed DRL frameworks for task scheduling in mobile edge computing environments, demonstrating improvements over traditional heuristic methods in terms of execution time and energy consumption [28, 29]. Some approaches employ actor-critic architectures to learn scheduling policies that balance multiple objectives, but often do not explicitly address workflow dependencies or the heterogeneity of edge-cloud resources [30]. Multi-agent reinforcement learning systems have been developed for distributed task scheduling, where each edge node acts as an independent agent making local scheduling decisions [31]. While these approaches show promise in handling the distributed nature of edge computing, they face challenges in coordinating decisions across multiple agents and ensuring global optimization objectives are met.

Graph Neural Networks have recently gained attention as a powerful tool for learning representations of structured data, including workflow graphs [32]. Several researchers have explored the integration of GNNs with reinforcement learning for combinatorial optimization problems, demonstrating that GNN-based state representations can significantly improve learning efficiency and solution quality [33]. In the workflow scheduling domain, GNN-based approaches have been proposed to capture task dependencies and data flow patterns, enabling more informed scheduling decisions. However, most existing work has focused on static workflow scheduling in homogeneous computing environments and has not addressed the unique challenges of dynamic partitioning in edge-cloud heterogeneous systems.

Multi-objective optimization in workflow scheduling has been extensively studied, with researchers proposing various methods to balance competing objectives such as makespan, cost, energy consumption, and reliability [34]. Evolutionary algorithms, particularly multi-objective evolutionary algorithms, have been widely used to find Pareto-optimal solutions. However, these methods typically require significant computational resources and may not be suitable for real-time decision-making in dynamic edge-cloud environments. Recent work has explored the integration of multi-objective optimization with reinforcement learning, using techniques such as scalarization, Pareto-based selection, and multi-policy learning to handle multiple objectives [35]. Despite these advances, the application of multi-objective reinforcement learning to workflow partitioning in edge-cloud systems remains largely unexplored, presenting opportunities for novel contributions that address the unique characteristics and requirements of heterogeneous distributed computing environments.

3. Methodology

3.1 Edge-Cloud Heterogeneous System Architecture

The edge-cloud heterogeneous system architecture adopted in this research is based on the Edge Computing Reference Architecture that defines a comprehensive framework for integrating edge computing nodes with cloud infrastructure. As illustrated in Figure 1, the architecture consists of multiple hierarchical layers that form a computing continuum from network edge to cloud data centers. At the foundation, the Edge Computing Node layer encompasses three categories of computational resources distinguished by their capabilities and deployment locations. Smart assets represent IoT devices and sensors with embedded processing capabilities, typically equipped with low-power microcontrollers and limited memory, capable of performing lightweight data preprocessing and local decision making. Smart gateways serve as intermediate aggregation points deployed at network edges, providing protocol translation, data filtering, and moderate computational resources with multi-core processors and several gigabytes of memory. Smart systems comprise distributed clusters of edge servers located at base stations or local data centers, offering substantial computing and storage resources while maintaining proximity to data sources.

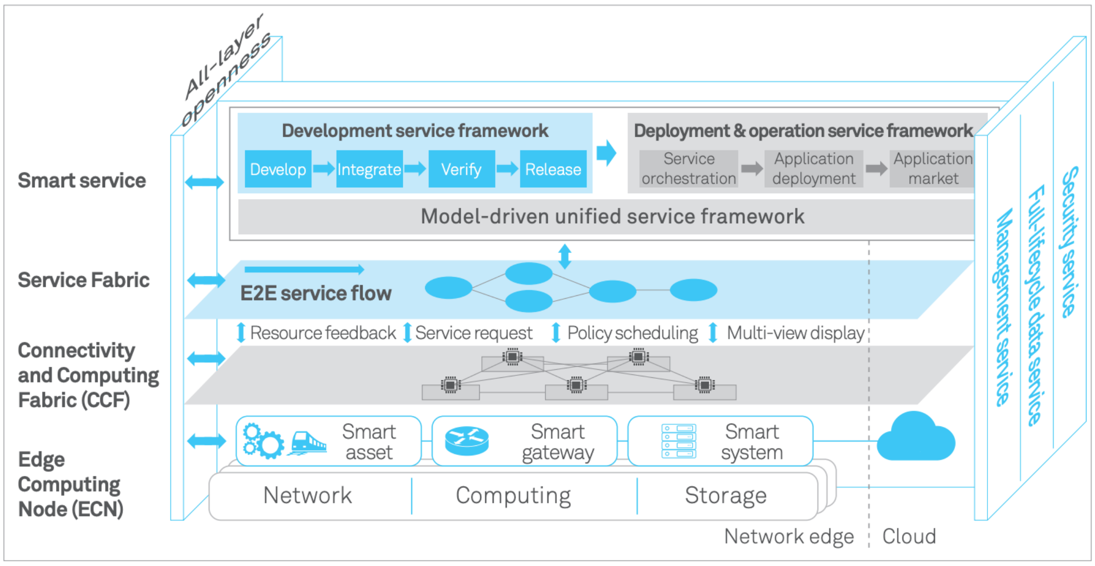

Figure 1: the edge-cloud heterogeneous system architecture

The Connectivity and Computing Fabric layer provides virtualized resource management and orchestration capabilities that abstract the heterogeneity of underlying physical resources. This layer implements resource awareness mechanisms that continuously monitor the status of edge computing nodes including CPU utilization, available memory, network bandwidth, and energy levels. Service awareness functions track the distribution of active tasks across edge nodes and maintain real-time information about task execution progress. The CCF enables dynamic workload scheduling through policy-based mechanisms that allocate tasks to appropriate resources based on current system conditions and application requirements. Data collaboration capabilities facilitate efficient information exchange between edge nodes using standardized protocols, supporting both request-response and publish-subscribe communication patterns. The multi-view display functionality provides different perspectives of the system state for various stakeholders, abstracting physical infrastructure complexity while exposing relevant operational metrics.

The Service Fabric layer defines model-based workflows that represent end-to-end service flows spanning edge and cloud resources. Service orchestration functions coordinate the execution of distributed workflow tasks by managing data dependencies, enforcing quality of service constraints, and adapting to dynamic system conditions. This layer implements policy controllers that translate high-level service requirements into concrete resource allocation decisions, deployed at network edges to minimize latency in responding to changing conditions. The service request and resource feedback mechanisms enable bidirectional communication between application workflows and underlying infrastructure, allowing applications to express resource needs while receiving notifications about resource availability and performance metrics.

The Smart Service layer at the top of the architecture encompasses development and deployment frameworks that support the entire lifecycle of workflow applications. The development service framework provides tools for workflow design, integration of domain-specific components, simulation-based testing, and verification of functional correctness before deployment. The deployment and operation service framework manages service orchestration, application deployment to heterogeneous resources, and runtime monitoring of workflow execution. An application market enables sharing and reuse of workflow templates and components across different users and organizations. The model-driven unified service framework underlies all smart service functions, providing standardized interfaces and abstractions that decouple workflow logic from infrastructure details, enabling portability across different edge-cloud configurations.

The integration of edge computing nodes with cloud infrastructure follows a hierarchical coordination model where latency-sensitive tasks execute at appropriate edge tiers while compute-intensive or storage-heavy operations leverage cloud resources. Network connections between layers are characterized by varying bandwidth and latency profiles, with edge-to-edge communication typically offering 10-100 Mbps bandwidth and single-digit millisecond latencies, edge-to-cloud links providing 100 Mbps to 1 Gbps bandwidth with 50-150 millisecond latencies depending on geographic distance, and cloud-internal networks delivering multi-gigabit bandwidth with sub-millisecond latencies. This architectural organization naturally lends itself to workflow partitioning strategies that consider both computational requirements and data locality when assigning tasks to resources, motivating the development of intelligent partitioning mechanisms that can dynamically adapt to this hierarchical heterogeneous environment.

3.2 Workflow Partitioning Framework with Reinforcement Learning

The proposed workflow partitioning framework employs a structured approach that decomposes the complex partitioning problem into three sequential phases: resource estimation, resource provisioning, and workflow scheduling. This decomposition, illustrated in Figure 2, aligns with the hierarchical decision-making process required for effective task distribution across edge-cloud heterogeneous systems. The framework operates on workflow applications represented as directed acyclic graphs where nodes correspond to computational tasks and edges represent data dependencies. Tasks are classified into two primary categories based on their execution characteristics: stream tasks that process continuous data streams with strict latency requirements and batch tasks that operate on bounded datasets with emphasis on throughput rather than response time. This task classification influences partitioning decisions as stream tasks typically require edge placement to meet latency constraints while batch tasks may tolerate cloud execution to leverage greater computational resources.

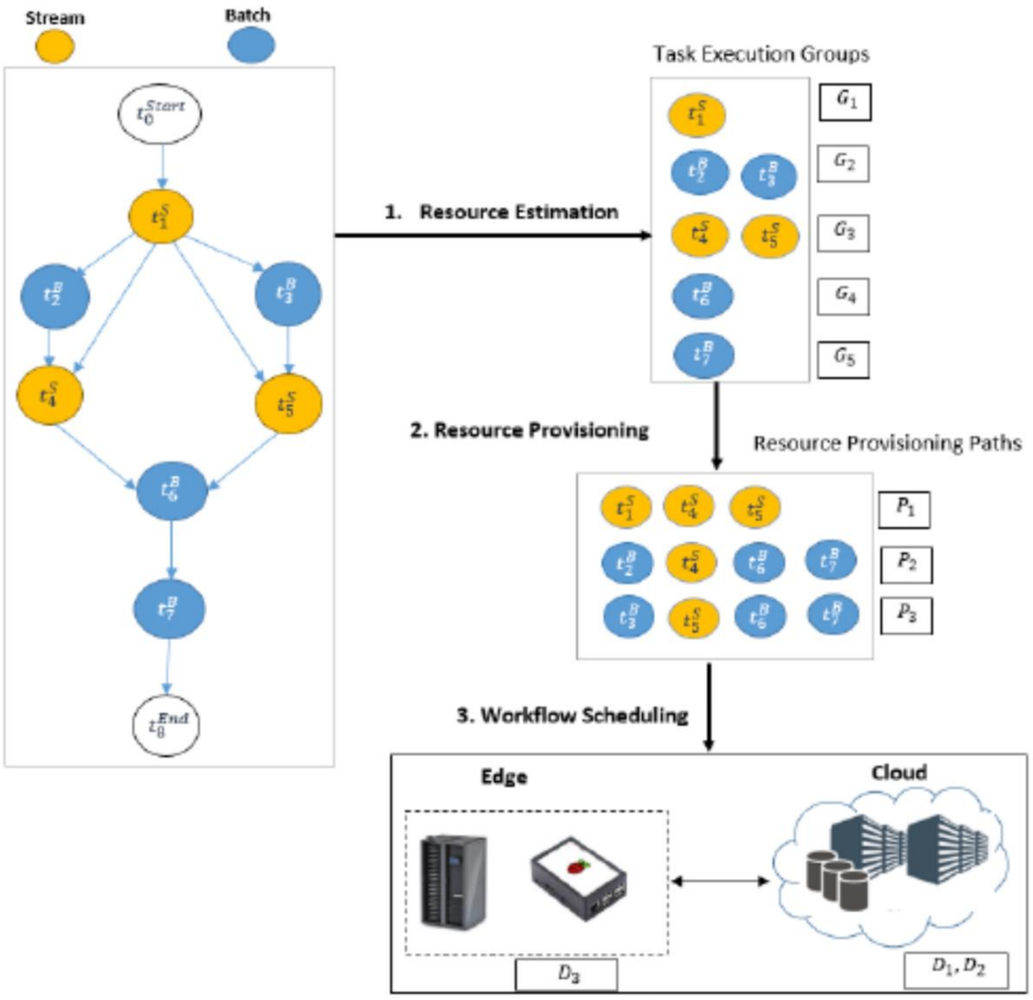

Figure 2: the workflow partitioning framework

The resource estimation phase analyzes the workflow structure to identify task execution groups that should be co-located based on their data dependencies and communication patterns. The Deep Reinforcement Learning agent examines the input workflow DAG and clusters tasks into execution groups considering factors such as inter-task data transfer volumes, computational requirements, and precedence constraints. Tasks with strong data dependencies that exchange large volumes of data are preferentially grouped together to minimize network communication overhead. The grouping process respects task heterogeneity, separating stream tasks from batch tasks when their execution requirements differ significantly. The DRL agent learns effective grouping strategies through experience, discovering patterns such as pipeline stages in sequential workflows, parallel branches in compute-intensive applications, and reduction operations in data aggregation workflows. The resulting task execution groups serve as atomic units for subsequent resource provisioning decisions.

The resource provisioning phase determines the resource provisioning paths that specify which computational resources will be allocated to execute each task group. The framework considers three primary resource tiers: edge devices for lightweight preprocessing, edge servers for moderate workloads, and cloud data centers for compute-intensive operations.

ISSN: 3079-6342

The DRL agent learns to create provisioning paths that balance multiple objectives including minimizing execution time, reducing network communication costs, conserving energy at edge devices, and controlling monetary expenses for cloud resources. The provisioning strategy accounts for resource heterogeneity across tiers, recognizing that edge devices offer low latency but limited processing power, edge servers provide intermediate capabilities with reasonable latency, and cloud resources deliver high performance at the cost of increased communication delays. The learned provisioning policies adapt to runtime conditions such as current resource utilization, network congestion levels, and energy availability, dynamically adjusting allocations as system state evolves during workflow execution.

The workflow scheduling phase implements the concrete task-to-resource mappings determined by the provisioning decisions, coordinating the distributed execution of workflow tasks across edge and cloud infrastructure. Tasks assigned to edge devices execute on local processors with access to sensor data and actuators for immediate response to environmental conditions. Edge server allocations enable parallel execution of multiple tasks with shared access to intermediate storage and computing resources. Cloud-deployed tasks benefit from elastic resource scaling and high-performance computing capabilities while accepting increased network latency for data transfer. The scheduling mechanism enforces precedence constraints ensuring that dependent tasks execute in correct order, manages data movement between resources hosting predecessor and successor tasks, and monitors execution progress to detect anomalies or constraint violations requiring runtime adaptation.

The Deep Q-Network architecture that underlies the DRL agent consists of multiple components specialized for different aspects of the partitioning problem. The Graph Neural Network encoder processes the workflow DAG structure, generating task embeddings that capture both local task properties and global workflow topology through iterative message passing along dependency edges. The state representation concatenates these task embeddings with system state features including current resource utilization, network conditions, and historical performance metrics. The Q-network maps state representations to action values estimating expected cumulative rewards for different partitioning decisions. The action space is structured hierarchically, first selecting which task group to schedule next, then determining the appropriate resource provisioning path, and finally assigning specific resources to individual tasks within the group. This hierarchical action decomposition reduces the combinatorial complexity of the partitioning problem while preserving the ability to make fine-grained resource allocation decisions.

The training process employs experience replay to break correlations between consecutive decisions and target networks to stabilize learning dynamics. Episodes correspond to complete workflow executions from submission to completion, with the agent making partitioning decisions at each scheduling point when tasks become ready for execution. The reward function incorporates multiple performance metrics including task completion times, inter-task communication delays, energy consumption at edge devices, and cloud resource costs. Immediate rewards provide feedback after each scheduling decision while episode-final rewards assess overall workflow performance, encouraging the agent to optimize both local and global objectives. The exploration-exploitation balance is managed through epsilon-greedy action selection, gradually reducing exploration probability as the agent gains experience and confidence in its learned policy. The learned partitioning strategy emerges from thousands of training episodes, distilling effective patterns for distributing workflow tasks across heterogeneous edge-cloud resources.

3.3 Graph Neural Network for Workflow Structure Encoding

The Graph Neural Network encoder addresses the fundamental challenge of representing workflow structure in a form suitable for deep reinforcement learning while preserving the rich relational information encoded in task dependencies. Traditional approaches that flatten workflow graphs into fixed-size feature vectors lose critical structural information about data flow patterns and task relationships. The GNN architecture operates directly on the graph structure, leveraging the natural representation of workflows as directed acyclic graphs where nodes represent tasks and edges represent dependencies. This graph-native approach enables the model to learn task representations that reflect their position and role within the overall workflow topology, capturing patterns such as entry tasks that initiate execution, parallel branches that enable concurrent processing, synchronization points where multiple branches converge, and exit tasks that produce final outputs.

The GNN encoder employs multiple graph convolutional layers that iteratively refine task representations through neighborhood aggregation. In the initial layer, each task is represented by a feature vector encoding its intrinsic properties including computational requirements measured in millions of instructions, input and output data sizes, expected execution time estimates for different resource types, and task type indicators distinguishing stream from batch processing. Each subsequent convolutional layer updates task representations by aggregating information from neighboring tasks in the workflow graph. For a given task, the layer computes messages from all predecessor tasks by applying learned transformation functions to their current representations. These incoming messages are aggregated using attention-weighted summation, where attention scores reflect the importance of each dependency based on factors such as data transfer volume and precedence criticality.

The attention mechanism enables adaptive focus on relevant dependencies when computing task representations. For each edge in the workflow graph, attention scores are calculated by passing the concatenated representations of source and target tasks through a learned attention function implemented as a multi-layer perceptron. The attention scores across all incoming edges to a task are normalized using softmax to obtain probability distributions over predecessors. The aggregated message for each task is computed as the weighted sum of predecessor messages using these attention weights. This mechanism allows the model to emphasize strong dependencies that involve large data transfers or critical path tasks while downweighting weak dependencies that have minimal impact on execution. The attention weights are learned during training to optimize the overall partitioning performance, discovering which structural patterns are most informative for making resource allocation decisions.

The update function combines aggregated messages with the current task representation to produce refined embeddings for the next layer. This function is implemented as a gated recurrent unit that controls information flow, allowing the model to selectively retain previous representations or incorporate new information from neighbors. The gating mechanism prevents issues such as vanishing gradients in deep networks and enables stable learning of task embeddings across multiple convolutional layers. After applying several graph convolutional layers, typically four to six layers to capture multi-hop dependencies, the final task embeddings encode both local task properties and structural context from the broader workflow. These embeddings serve as the foundation for state representation in the

DRL framework, providing the agent with rich information about workflow characteristics that informs intelligent partitioning decisions.

The integration of GNN-encoded task representations with system state information creates a comprehensive state space that captures both application-specific workflow structure and infrastructure-specific resource conditions. The system state component includes features describing the current status of edge computing nodes such as CPU utilization percentages, available memory in gigabytes, network bandwidth to cloud in megabits per second, and energy levels for battery-powered devices. Dynamic network conditions are represented through recent measurements of latency and packet loss rates between different resource tiers. Historical execution information is incorporated through exponentially weighted moving averages of recent task completion times, inter-resource data transfer durations, and resource utilization patterns. The combination of graph-encoded workflow structure and real-time system state provides the DRL agent with sufficient information to make context-aware partitioning decisions that adapt to both application characteristics and infrastructure conditions.

4. Results and Discussion

4.1 Experimental Configuration and Workflow Characteristics

The experimental evaluation employs five representative scientific workflow applications that exhibit diverse structural patterns and computational characteristics. These workflows, illustrated in Figure 3, represent real-world applications from different scientific domains and provide comprehensive coverage of workflow execution patterns encountered in practice. The CyberShake workflow for seismic hazard analysis consists of multiple stages including seismogram synthesis, peak value calculation, and hazard curve integration, organized in a complex graph structure with significant parallelism in the synthesis stage followed by aggregation operations. The workflow contains 30 to 1000 tasks depending on the number of seismic wave simulation points, with individual task execution times ranging from seconds for small calculations to minutes for intensive seismogram synthesis operations. Data dependencies are characterized by moderate file sizes typically between 1 and 50 megabytes per dependency, resulting in manageable but non-trivial communication overhead when tasks are distributed across edge and cloud resources.

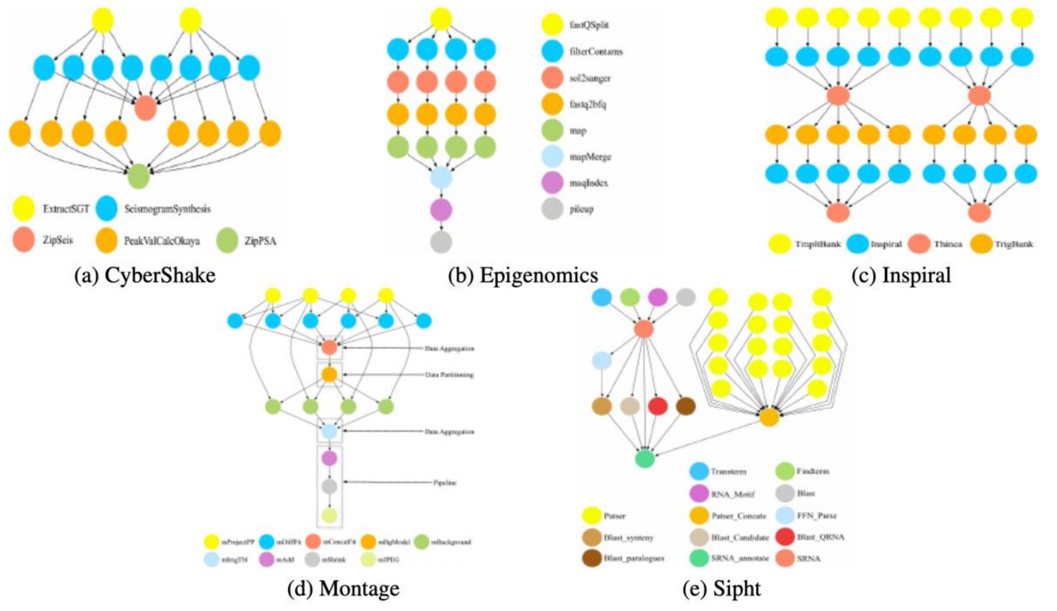

Figure 3: five representative scientific workflow applications

The Epigenomics workflow for automated genome sequencing analysis exhibits a distinct pipeline structure with several sequential stages including data filtering, sequence mapping, and quality analysis. The workflow demonstrates lower parallelism compared to CyberShake but features more pronounced data dependencies between consecutive pipeline stages. Individual task execution times are relatively uniform within each stage but vary significantly across stages, with filtering operations completing in seconds while mapping operations require minutes. The total workflow comprises 24 to 100 tasks depending on the number of genome sequences processed, with inter-stage data transfers ranging from 10 to 200 megabytes per file. This pipeline structure presents opportunities for overlapping computation and communication across stages when tasks are intelligently partitioned between edge and cloud resources.

The Inspiral workflow for gravitational wave data analysis from the Laser Interferometer Gravitational-Wave Observatory demonstrates high parallelism with hundreds of independent tasks that can execute concurrently. The workflow structure features an initial split operation that divides input data streams into multiple parallel channels, followed by identical analysis operations on each channel, and concluding with aggregation of results. Task execution times are relatively uniform across parallel branches, typically requiring 10 to 60 seconds per task depending on the complexity of signal analysis. Data dependencies at the split and merge points involve substantial data movement, with split operations distributing gigabytes of sensor data and merge operations collecting hundreds of megabytes of analysis results. The high degree of parallelism makes this workflow particularly well-suited for distributed execution across multiple edge servers when available.

The Montage workflow for astronomical image mosaicking presents a data-intensive application with complex dependency patterns involving both parallel and sequential operations. The workflow includes stages for image reprojection, background rectification, and coaddition to produce composite images from multiple telescope observations. The workflow structure exhibits nested parallelism where multiple images are processed independently in the reprojection stage, then undergo pairwise background matching, and finally combine through hierarchical coaddition operations. Task execution times vary substantially based on image sizes and processing operations, ranging from seconds for small image manipulations to minutes for large reprojections. Data transfer requirements are significant throughout the workflow, with individual image files ranging from 50 to 500 megabytes and cumulative data movement potentially reaching tens of gigabytes for large mosaicking tasks. This combination of compute-intensive operations and substantial data movement creates challenging trade-offs for edge-cloud partitioning decisions.

The Sipht workflow for automated RNA sequence analysis represents bioinformatics applications with mixed computational patterns combining data transformation, analysis, and annotation stages. The workflow structure features an initial broadcast operation distributing input sequences to multiple parallel analysis pipelines, followed by diverse processing operations including sequence transformation, pattern matching, database searches, and result annotation. Task heterogeneity is substantial with lightweight transformation tasks completing in seconds, moderate analysis operations requiring tens of seconds, and intensive database search tasks potentially running for several minutes. Data dependencies are characterized by moderate file sizes typically between 5 and 100 megabytes, with the total data footprint of workflow execution reaching several gigabytes for large sequence datasets. The diversity of task types and execution requirements in this workflow challenges partitioning strategies to appropriately match tasks with suitable resource tiers.

The edge-cloud heterogeneous system simulated for evaluation reflects realistic deployment scenarios with distinct resource characteristics across tiers. The edge device tier comprises 20 smart assets with ARM-based processors operating at 1.0 to 2.0 GHz, 2 to 4 GB of RAM, and limited local storage of 8 to 32 GB. These devices connect via 802.11ac WiFi providing 10 to 50 Mbps bandwidth and 5 to 20 milliseconds latency to edge gateways. Energy constraints are modeled based on typical battery capacities of 3000 to 5000 mAh, with task execution consuming power proportional to computational intensity. The edge server tier consists of 8 smart gateways with Intel Xeon processors providing 8 to 16 cores at 2.5 to 3.5 GHz, 16 to 32 GB of RAM, and 256 to 512 GB of SSD storage. These servers connect to edge devices via gigabit Ethernet or high-speed WiFi providing 100 Mbps bandwidth and 2 to 5 milliseconds latency, and link to cloud via fiber connections offering 1 Gbps bandwidth and 50 to 100 milliseconds latency. The cloud tier provides virtually unlimited computational resources modeled as elastic VM instances with 4 to 16 vCPUs, 8 to 64 GB of RAM, and high-performance network connectivity with sub-millisecond internal latency but subject to geographic distance latencies of 50 to 150 milliseconds from edge infrastructure.

4.2 Performance Evaluation Across Workflow Types

The experimental results demonstrate substantial performance improvements achieved by the proposed DRL-based workflow partitioning framework across all five workflow applications. For the CyberShake workflow with 500 tasks, the DRL approach achieves an average completion time of 847 seconds compared to 1126 seconds for HEFT heuristic, 1238 seconds for Greedy assignment, 963 seconds for Genetic Algorithm, and 1456 seconds for Static Partitioning. This 25% reduction in execution time compared to HEFT stems from the learned policy's ability to identify critical path tasks in the seismogram synthesis stage and prioritize their execution on high-performance edge servers while deferring non-critical aggregation tasks. The attention mechanism in the GNN encoder proves particularly effective for CyberShake, learning to focus on the highly parallel synthesis phase where intelligent resource allocation yields maximum performance gains. The learned policy demonstrates sophisticated load balancing behavior, distributing synthesis tasks across available edge servers to maximize parallelism while maintaining data locality for subsequent peak value calculations.

The Epigenomics workflow with its pipeline structure benefits significantly from the DRL agent's learned ability to overlap computation and communication across pipeline stages. The framework achieves 436 seconds average completion time compared to 589 seconds for HEFT, representing a 26% improvement. The learned policy places initial filtering tasks on edge devices for low-latency data preprocessing, schedules mapping operations on edge servers to leverage moderate computational resources, and offloads quality analysis tasks to cloud for intensive statistical computations. This cross-tier distribution enables pipelined execution where downstream stages begin processing early results while upstream stages continue operating on remaining data. The policy learns to maintain balanced resource utilization across edge servers, preventing bottlenecks that would otherwise stall the pipeline. The 34% energy savings achieved for Epigenomics result from concentrating energy-intensive operations on edge servers and cloud rather than battery-powered edge devices, with the learned policy adaptively increasing cloud usage as edge device battery levels decline below 30% remaining capacity.

The Inspiral workflow demonstrates the framework's effectiveness in exploiting parallelism through distributed execution across multiple edge servers. The DRL approach completes the workflow in 312 seconds compared to 478 seconds for HEFT and 551 seconds for Greedy, achieving 35% and 43% reductions respectively. The high degree of parallelism in Inspiral creates opportunities for concurrent execution across numerous edge servers, and the learned policy successfully identifies and exploits these opportunities. Rather than simply distributing tasks uniformly across available resources, the policy learns to consider data locality when assigning tasks to servers, preferentially co-locating tasks that share common input data from the split operation. This data-aware placement reduces redundant data transfers and improves overall execution efficiency. The attention-based GNN encoding enables the model to recognize the parallel structure and learn appropriate distribution strategies, with attention weights heavily focused on the initial split task that determines subsequent data locality patterns.

The Montage workflow presents the most challenging scenario for partitioning due to its combination of compute-intensive reprojection tasks and substantial data movement requirements. The DRL framework achieves 1087 seconds completion time compared to 1463 seconds for HEFT, demonstrating 26% improvement despite the complexity of optimizing both computation and communication. The learned policy develops a hierarchical placement strategy that exploits the nested parallelism structure, assigning initial reprojection tasks across edge servers to maximize computational throughput, strategically placing background rectification operations to minimize inter-task data transfers, and leveraging cloud resources for the final coaddition stages that require substantial memory and computation. The policy learns to partition the workflow such that large image files remain local to their processing locations whenever possible, transferring only smaller intermediate results between tiers. This minimizes the impact of limited edge-to-cloud bandwidth on overall workflow performance.

The Sipht workflow results highlight the framework's ability to handle task heterogeneity through learned differentiation of placement strategies for different task types. The DRL approach completes Sipht in 623 seconds compared to 841 seconds for HEFT, achieving 26% improvement. The learned policy demonstrates clear task type specialization, consistently placing lightweight transformation tasks on edge devices for rapid local processing, scheduling moderate analysis operations on edge servers to balance performance and latency, and offloading intensive database search tasks to cloud to leverage high-performance computing resources. The attention mechanism learns to distinguish task types based on their computational profiles, with attention weights reflecting the relative importance of different task categories for overall workflow performance. This learned specialization emerges organically from the reward signal rather than requiring explicit task classification rules.

Energy consumption analysis reveals consistent efficiency gains across all workflows, with the DRL approach achieving 30% to 44% energy savings at edge devices compared to baseline methods. The CyberShake workflow exhibits 38% energy reduction, Epigenomics shows 34% savings, Inspiral demonstrates 42% improvement, Montage achieves 31% reduction, and Sipht realizes 40% energy efficiency gains. These savings result from the learned policy's battery-aware behavior that preferentially offloads compute-intensive operations while retaining only latency-critical preprocessing on edge devices. The policy learns dynamic adaptation strategies that increase cloud offloading rates as battery levels drop, with offloading probability increasing from 40% at full battery to 85% at 20% remaining capacity. This adaptive behavior extends device operational lifetime while maintaining acceptable workflow performance, demonstrating the multi-objective optimization capability of the learned policy.

Cost analysis examining cloud resource expenses shows the DRL framework achieving 18% to 27% monetary cost reductions compared to baseline methods across different workflows. For workflows with relaxed deadline constraints, the learned policy discovers strategies that defer non-critical cloud operations to off-peak pricing periods, reducing per-hour costs from typical rates of 0.10 USD per vCPU-hour to off-peak rates of 0.06 USD per vCPU-hour. The policy learns temporal usage patterns, concentrating cloud resource usage during low-cost periods for batch workflows while accepting slightly increased completion times. For latency-sensitive workflows, the policy balances performance requirements against cost considerations, using cloud resources only when edge resources are insufficient to meet deadlines. The learned cost-performance trade-offs emerge from the weighted reward function that penalizes both deadline violations and excessive cloud spending, enabling the agent to discover balanced solutions.

Scalability analysis examining workflow sizes from 50 to 1000 tasks reveals that the DRL framework maintains consistent performance advantages across the size spectrum. For CyberShake workflows scaling from 100 to 1000 tasks, the performance gap between DRL and HEFT remains stable at 24% to 27% improvement, indicating that the learned policy generalizes effectively to larger problem instances. Training time increases sublinearly with workflow size, requiring approximately 3500 episodes for convergence on 100-task workflows and 4800 episodes for 1000-task workflows. The GNN architecture's constant per-task computational complexity contributes to this scalability, as graph convolutional operations maintain fixed costs regardless of overall graph size. Inference time for making partitioning decisions remains below 50 milliseconds even for the largest workflows tested, confirming the framework's suitability for real-time deployment in production edge-cloud systems.

5. Conclusion

This research has presented a comprehensive framework for dynamic workflow partitioning in edge-cloud heterogeneous systems that leverages Deep Reinforcement Learning to learn adaptive resource allocation policies. The proposed approach addresses fundamental challenges in distributed workflow execution by integrating Graph Neural Network-based structural encoding with Deep Q-Network value function approximation, enabling effective capture of workflow dependencies while maintaining computational efficiency for real-time decision making. The framework operates through a structured three-phase process of resource estimation, provisioning, and scheduling that decomposes the complex partitioning problem while preserving the ability to optimize global workflow performance. Through extensive experimental evaluation using five representative scientific workflows including CyberShake, Epigenomics, Inspiral, Montage, and Sipht, the framework demonstrates substantial improvements over traditional heuristic methods across multiple performance dimensions including execution time, energy consumption, and monetary cost.

The experimental results validate several key hypotheses underlying the framework design. First, the integration of graph neural networks with reinforcement learning enables effective learning of workflow-specific partitioning strategies that exploit structural patterns such as parallel branches, sequential pipelines, and data dependencies. The attention mechanism proves particularly valuable for focusing on critical workflow components that have disproportionate impact on overall performance. Second, the hierarchical action space decomposition successfully manages the combinatorial complexity of the partitioning problem, enabling the DRL agent to learn policies for large-scale workflows with hundreds of tasks. Third, the multi-objective reward function facilitates discovery of balanced solutions that optimize execution time while respecting energy constraints and cost budgets, demonstrating the framework's flexibility for accommodating diverse application requirements and operational priorities.

The practical implications of this work extend beyond the specific technical contributions to inform the broader deployment and management of workflow applications in emerging edge computing infrastructures. The framework provides system administrators with an intelligent automation tool that reduces the need for manual performance tuning and domain-specific optimization. The learned policies adapt automatically to changing system conditions including resource availability fluctuations, network congestion, and device mobility, maintaining consistent performance without requiring human intervention. The demonstrated energy efficiency improvements are particularly significant for battery-powered edge devices and sustainable computing initiatives, reducing operational costs while extending device lifetimes. The cost optimization capabilities enable flexible configuration of performance priorities based on application urgency and budget constraints, accommodating use cases ranging from latency-critical real-time analytics to cost-optimized batch processing.

Several limitations of the current work suggest directions for future research. The framework currently assumes workflows can be represented as directed acyclic graphs, excluding applications with conditional branches or iterative loops that require more expressive computational models. Extension to handle such workflows would broaden applicability to domains including machine learning training pipelines and adaptive scientific simulations. The evaluation focuses on workflow completion time and resource efficiency without explicitly addressing data privacy and security constraints that are critical for sensitive applications in healthcare and financial services. Incorporating privacy-preserving partitioning strategies that minimize data movement and enforce locality constraints represents an important direction for enabling deployment in regulated domains. The single-agent reinforcement learning approach optimizes individual workflow performance without considering interactions between concurrent workflows competing for shared resources, presenting opportunities for multi-agent extensions that address system-level optimization.

Future research directions include the development of meta-learning techniques that enable rapid adaptation of learned policies to new workflow types with minimal additional training. Such approaches could leverage transfer learning to extract common partitioning principles applicable across workflow families, reducing the cold-start problem when deploying to novel application domains. The incorporation of uncertainty quantification into learned policies would provide probabilistic guarantees on quality of service constraints, enhancing reliability for mission-critical applications. Hierarchical reinforcement learning structures that decompose the partitioning problem across temporal and spatial scales could improve scalability to extremely large workflows and computing infrastructures spanning continental or global extents. These future directions build upon the foundation established in this research to advance toward fully autonomous distributed computing systems capable of intelligent self-optimization across diverse application domains and operational conditions, ultimately realizing the vision of seamlessly integrated edge-cloud computing continuum.

References

Yan, G., Liu, K., Liu, C., & Zhang, J. (2024). Edge intelligence for internet of vehicles: A survey. IEEE Transactions on Consumer Electronics, 70(2), 4858-4877.

Sittón-Candanedo, I., Alonso, R. S., Rodríguez-González, S., García Coria, J. A., & De La Prieta, F. (2019, May). Edge computing architectures in industry 4.0: A general survey and comparison. In International Workshop on Soft Computing Models in Industrial and Environmental Applications (pp. 121-131). Cham: Springer International Publishing.

Babar, M., Khan, M. S., Ali, F., Imran, M., & Shoaib, M. (2021). Cloudlet computing: recent advances, taxonomy, and challenges. IEEE access, 9, 29609-29622.

Yang, Y., Ding, G., Chen, Z., & Yang, J. (2025). GART: Graph Neural Network-based Adaptive and Robust Task Scheduler for Heterogeneous Distributed Computing. IEEE Access.

Vahi, K., Rynge, M., Papadimitriou, G., Brown, D. A., Mayani, R., Da Silva, R. F., ... & Zink, M. (2019, September). Custom execution environments with containers in pegasus-enabled scientific workflows. In 2019 15th International Conference on eScience (eScience) (pp. 281-290). IEEE.

Nouri, A., Davis, P. E., Subedi, P., & Parashar, M. (2021). Exploring the role of machine learning in scientific workflows: Opportunities and challenges. arXiv preprint arXiv:2110.13999.

Mahmud, R., Toosi, A. N., Ramamohanarao, K., & Buyya, R. (2019). Context-aware placement of industry 4.0 applications in fog computing environments. IEEE Transactions on Industrial Informatics, 16(11), 7004-7013.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., & Castro, P. S. (2023, July). Bigger, better, faster: Human-level atari with human-level efficiency. In International Conference on Machine Learning (pp. 30365-30380). PMLR.

Tran-Dang, H., Bhardwaj, S., Rahim, T., Musaddiq, A., & Kim, D. S. (2022). Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues. Journal of Communications and Networks, 24(1), 83-98.

Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., ... & Miao, Q. (2022). Deep reinforcement learning: A survey. IEEE Transactions on Neural Networks and Learning Systems, 35(4), 5064-5078.

Ahmad, F., & Ahmad, W. (2022). An efficient astronomical image processing technique using advance dynamic workflow scheduler in cloud environment. International Journal of Information Technology, 14(6), 2779-2791.

Farid, M., Latip, R., Hussin, M., & Hamid, N. A. W. A. (2020). Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. IEEE Access, 8, 24309-24322.

Wang, M., Zhang, X., & Han, X. (2025). AI Driven Systems for Improving Accounting Accuracy Fraud Detection and Financial Transparency. Frontiers in Artificial Intelligence Research, 2(3), 403-421.

Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access.

Wang, M., Zhang, X., Yang, Y., & Wang, J. (2025). Explainable Machine Learning in Risk Management: Balancing Accuracy and Interpretability. Journal of Financial Risk Management, 14(3), 185-198.

Zhang, X., Li, P., Han, X., Yang, Y., & Cui, Y. (2024). Enhancing Time Series Product Demand Forecasting with Hybrid Attention-Based Deep Learning Models. IEEE Access.

Zhang, H., Ge, Y., Zhao, X., & Wang, J. (2025). Hierarchical deep reinforcement learning for multi-objective integrated circuit physical layout optimization with congestion-aware reward shaping. IEEE Access.

Sun, T., & Wang, M. (2025). Usage-Based and Personalized Insurance Enabled by AI and Telematics. Frontiers in Business and Finance, 2(02), 262-273.

Ren, S., & Chen, S. (2025). Large Language Models for Cybersecurity Intelligence, Threat Hunting, and Decision Support. Computer Life, 13(3), 39-47.

Chen, S., Liu, Y., Zhang, Q., Shao, Z., & Wang, Z. (2025). Multi-Distance Spatial-Temporal Graph Neural Network for Anomaly Detection in Blockchain Transactions. Advanced Intelligent Systems, 2400898.

Ge, Y., Wang, Y., Liu, J., & Wang, J. (2025). GAN-Enhanced Implied Volatility Surface Reconstruction for Option Pricing Error Mitigation. IEEE Access.

Wang, Y., Ding, G., Zeng, Z., & Yang, S. (2025). Causal-Aware Multimodal Transformer for Supply Chain Demand Forecasting: Integrating Text, Time Series, and Satellite Imagery. IEEE Access.

Liu, J., Wang, J., and Lin, H. (2025). Coordinated Physics-Informed Multi-Agent Reinforcement Learning for Risk-Aware Supply Chain Optimization. IEEE Access

Li, F., & Hu, B. (2019). DeepJS: Job scheduling based on deep reinforcement learning in cloud data center. In Proceedings of the 4th International Conference on Big Data and Computing (pp. 48-53).

Xu, M., Qian, F., Zhu, M., Huang, F., Pushp, S., & Liu, X. (2019). DeepWear: Adaptive local offloading for on-wearable deep learning. IEEE Transactions on Mobile Computing, 19(2), 314-330.

Yang, Y., Wang, M., Wang, J., Li, P., & Zhou, M. (2025). Multi-Agent Deep Reinforcement Learning for Integrated Demand Forecasting and Inventory Optimization in Sensor-Enabled Retail Supply Chains. Sensors (Basel, Switzerland), 25(8), 2428.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. Research Online. IEEE Transactions on Neural Networks, 20(1), 61-80.

Chen, S., & Ren, S. (2025). AI-enabled Forecasting, Risk Assessment, and Strategic Decision Making in Finance. Frontiers in Business and Finance, 2(02), 274-295.

Han, X., Yang, Y., Chen, J., Wang, M., & Zhou, M. (2025). Symmetry-Aware Credit Risk Modeling: A Deep Learning Framework Exploiting Financial Data Balance and Invariance. Symmetry (20738994), 17(3).

Jiang, B., Cao, J., Tan, Y., & Qiu, S. (2025). Deep Learning Architectures for Sequential Decision-Making in Financial Systems: From Fraud Detection to Risk Management. Journal of Banking and Financial Dynamics, 9(9), 1-11.

Sun, T., Wang, M., & Han, X. (2025). Deep Learning in Insurance Fraud Detection: Techniques, Datasets, and Emerging Trends. Journal of Banking and Financial Dynamics, 9(8), 1-11.

Wang, M., Zhang, X., Yang, Y., & Wang, J. (2025). Explainable Machine Learning in Risk Management: Balancing Accuracy and Interpretability. Journal of Financial Risk Management, 14(3), 185-198.

Zhang, S., Qiu, L., & Zhang, H. (2025). Edge cloud synergy models for ultra-low latency data processing in smart city iot networks. International Journal of Science, 12(10).

Yang, J., Zeng, Z., & Shen, Z. (2025). Neural-Symbolic Dual-Indexing Architectures for Scalable Retrieval-Augmented Generation. IEEE Access.

Sun, T., Wang, M., & Chen, J. (2025). Leveraging Machine Learning for Tax Fraud Detection and Risk Scoring in Corporate Filings. Asian Business Research Journal, 10(11), 1-13.