using signature analysis. The method is integrated into the SDN control plane using CBQ and WRED mechanisms for adaptive queue management. Experimental studies in the Mininet + Floodlight environment confirmed that the combined Hurst–DPI approach provides an increase in attack detection accuracy up to 94%, a reduction in response time by 35%, and a reduction in false positives by 67% compared to traditional methods. The proposed algorithm allows to increase the fault tolerance of SDN networks and maintain the quality of service of critical services in the event of DDoS load.
Keywords: *software-defined networks, DDoS, Hurst index, DPI, QoS, fault tolerance.*

UDC: 004.032.26:004.75 **DOI: 10.18372/2073-4751.83.20552**

[1]Mukhin V.Ye.,
orcid.org/0000-0002-1206-9131,
[1]Kulyk V.O.,
orcid.org/0000-0003-3833-1529,
[2]Yaroviy O.V.,
orcid.org/0000-0002-3889-5730
[2]Kutsenko I.S.
orcid.org/0009-0005-7549-0643

COMPARATIVE ANALYSIS OF CONVOLUTIONAL AND MULTILAYER PERCEPTRON NEURAL NETWORKS FOR RESOURCE ALLOCATION IN DISTRIBUTED COMPUTING SYSTEMS

[1]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
[2]Scientific and Methodical Center of the Defence Intelligence Research Institute, Kyiv.

e-mail: v.mukhin@kpi.ua,
e-mail: getem13@ukr.net,

1. Introduction

1.1.Problem Context and Motivation

Distributed computing systems increasingly rely on heterogeneous nodes with varying capabilities, ranging from high-performance servers to resource-constrained edge devices. Efficient task allocation in such environments requires sophisticated mechanisms capable of matching task requirements to node characteristics while accounting for dynamic system conditions. Traditional scheduling algorithms employ fixed rules and heuristics that prove unable to adapt to changing workload patterns, network conditions, and resource availability. Machine learning approaches offer data-driven solutions that learn from historical patterns and node telemetry, enabling adaptive decision-making in complex distributed environments. Neural networks have emerged as particularly promising tools for capturing complex relationships between node attributes and task execution success. However, architecture selection significantly impacts not only prediction performance but also computational efficiency, training requirements, and deployment feasibility in production environments. Despite growing interest in AI-based scheduling, limited research exists comparing different neural architectures specifically for resource allocation tasks. This study addresses this gap through systematic comparison of Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) architectures for node suitability classification in distributed computing systems. [15]

1.2. Research Objectives and Scope

The primary goal of this research is to evaluate CNN and MLP performance for

binary node suitability classification across varying dataset sizes and system conditions. The study assesses accuracy, consistency, and computational efficiency metrics to provide comprehensive understanding of each architecture's strengths and limitations. This investigation examines how architecture choice affects model behavior with both limited and abundant training data, reflecting practical constraints faced during initial deployment versus mature system operation. The research provides practical recommendations for distributed systems practitioners selecting neural architectures for resource scheduling implementations. The study focuses on binary classification using four standard node attributes: performance (computational capacity), security level (protection mechanisms), baud rate (communication speed), and reliability (historical uptime). Experiments employ controlled synthetic datasets to isolate architecture effects from data collection artifacts and system-specific biases. The article structure proceeds as follows: Section 2 reviews related work and formulates the classification problem; Section 3 describes the proposed MLP and CNN architectures; Section 4 presents experimental setup and results; Section 5 discusses findings and practical implications; Section 6 concludes with key insights and future research directions. [16]

2. Related Work and Problem Formulation

2.1. Traditional and AI-Based Resource Allocation

Classical resource allocation methods in distributed systems include centralized scheduling algorithms, load balancing techniques, and heuristic-based approaches that rely on predefined rules and static priority schemes [1-3]. Recent years have witnessed a significant shift toward neural network-based approaches for adaptive task distribution, driven by the ability of deep learning models to capture complex patterns from historical execution data and real-time node telemetry [4-6]. Deep learning architectures enable learning intricate relationships between node characteristics, workload patterns, and task execution outcomes without requiring explicit feature engineering or domain-specific rule formulation. Reinforcement learning techniques have been successfully applied for dynamic offloading decisions in edge computing environments, where agents learn optimal policies through interaction with the distributed system. However, most prior work in AI-based scheduling adopts a single neural architecture without systematic justification or comparative evaluation of alternative designs. This gap is particularly evident in the domain of tabular resource data, where the suitability of different neural architectures remains under-explored. The present work contributes empirical evidence for architecture selection in this domain, providing systematic comparison of CNN and MLP approaches specifically tailored to node classification problems in distributed computing environments. [1, 2]

2.2. Problem Statement

The node suitability classification problem can be formally stated as follows: given current node attributes, classify each node as suitable or unsuitable for executing a specific task. The input consists of four node features that characterize operational capabilities and reliability: performance representing computational capacity measured in processing units, security indicating the protection level of implemented security mechanisms, baud rate quantifying communication speed for data transfer operations, and reliability reflecting historical uptime and availability patterns. The output is a binary label indicating node suitability for task assignment, with positive classification signifying that the node meets all necessary requirements for successful task execution. Training data for the classification model is collected from continuous system monitoring, where node attributes are periodically sampled and labeled based on subsequent task execution outcomes. The task is formulated as a supervised binary classification problem, where the neural network learns a mapping

from the four-dimensional attribute space to the binary decision space. The trained model must demonstrate robust generalization to unseen node states encountered during deployment, as the distributed system continuously evolves with changing workloads, network conditions, and resource availability patterns. [3, 17]

3. Proposed Neural Network Architectures

3.1. System Overview

The proposed node classification system implements an end-to-end pipeline consisting of four primary stages: monitoring agents collect node attributes from distributed computing nodes, data preprocessing normalizes attribute values to the [0,1] range to ensure consistent neural network input, the trained classifier performs inference to predict suitability probability for each candidate node, and the scheduler utilizes these predictions to assign incoming tasks to the highest-scoring available nodes. Node attributes are collected in real-time through continuous monitoring infrastructure that samples performance metrics, security configurations, communication capabilities, and reliability statistics at regular intervals. The trained neural network classifier processes these normalized attributes and produces probability scores indicating the likelihood of successful task execution on each node. The scheduling system leverages these predictions by maintaining a ranked list of nodes, enabling rapid task assignment decisions that optimize resource utilization while maintaining quality-of-service requirements. [2, 18, 19]

3.2. Multilayer Perceptron Architecture

The MLP architecture is designed with the rationale that fully connected layers can capture all possible interactions between node attributes without imposing structural assumptions on the data. The network topology consists of an input layer accepting four normalized features, followed by three hidden layers with progressively decreasing dimensions: the first hidden layer contains 64 neurons with ReLU activation, the second contains 32 neurons with ReLU activation, and the third contains 16 neurons with ReLU activation, culminating in an output layer with a single neuron using sigmoid activation to produce binary classification probabilities. Dropout regularization with probability 0.3 is applied after the first and second hidden layers to prevent overfitting by randomly deactivating neurons during training.

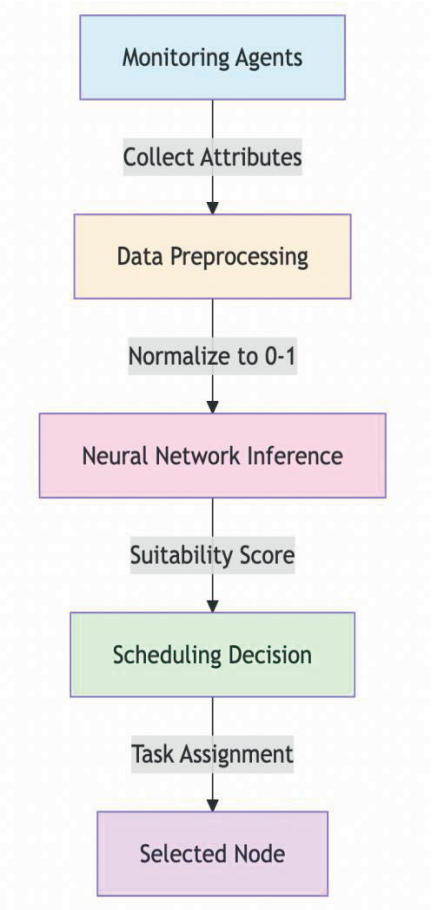

Figure 1: System architecture pipeline showing the flow from monitoring through preprocessing, neural network inference, and scheduling decision to final task assignment.

The complete architecture contains approximately 5,500 trainable parameters, calculated as the sum of weight matrices and bias vectors across all layers. Training employs the Adam optimizer with learning rate 0.001, binary cross-entropy loss function appropriate for binary classification, batch size of 32 samples, and early stopping with patience of 15 epochs to halt training when validation performance plateaus. The combination of dropout regularization and early stopping

effectively prevents overfitting while enabling the model to learn complex attribute interactions. The relatively simple architecture facilitates rapid training convergence and efficient deployment in resource-constrained environments where inference latency is critical. [2, 4, 20]

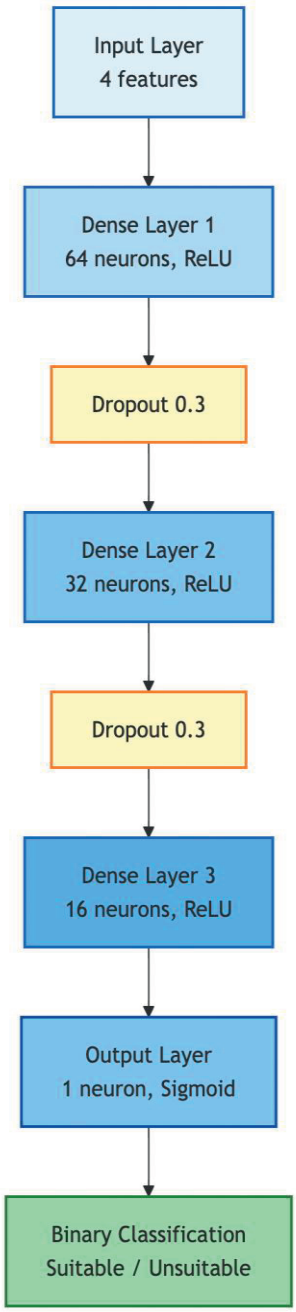

Figure 2: MLP architecture diagram showing layer configuration with dimensions, activation functions, and regularization techniques. Total parameters: ~5,500.

3.3. Convolutional Neural Network Architecture

The CNN architecture is designed to perform hierarchical feature extraction through convolution operations that identify local patterns within the attribute sequence.

The network begins by reshaping the four-dimensional input vector into shape (4,1) to enable one-dimensional convolution, followed by the first convolutional layer with 64 filters and kernel size 2, batch normalization for training stability, the second convolutional layer with 32 filters and kernel size 2, another batch normalization layer, a flattening operation to convert feature maps into a one-dimensional vector, two fully connected layers with 64 and 32 neurons respectively using ReLU activation, dropout regularization with probability 0.4 applied after the first dense layer, and an output neuron with sigmoid activation for binary classification.

The complete CNN architecture contains approximately 3,800 trainable parameters, fewer than the MLP due to parameter sharing inherent in convolutional operations where the same filter weights are applied across different positions in the input.

Training configuration remains identical to the MLP (Adam optimizer with learning rate 0.001, binary cross-entropy loss, batch size 32, early stopping with patience 15) to ensure fair comparison between architectures. The convolutional layers attempt to extract local patterns and dependencies from sequential arrangement of node attributes, while batch normalization stabilizes training dynamics by reducing internal covariate shift. Despite having fewer parameters due to weight sharing in convolutional operations, the CNN's inductive bias toward spatial locality may not align optimally with the unordered nature of tabular node attributes. [5]

4. Experimental Setup and Results

4.1. Dataset and Methodology

The experimental evaluation employs five synthetic datasets containing 100, 500,

1000, 1500, and 2000 instances respectively, designed to assess architecture performance across varying data availability conditions. Each dataset instance consists of four node attributes and one binary suitability label, representing realistic distributed computing scenarios. Node attributes are generated from appropriate probability distributions: performance values follow a uniform distribution simulating heterogeneous computational capabilities, security levels are drawn from a beta distribution reflecting varying protection implementations, baud rates follow a normal distribution representing typical network bandwidth variations, and reliability scores use a beta distribution modeling historical uptime patterns. Binary labels are assigned through a weighted threshold function that combines the four attributes with configurable weights, with 10% label noise introduced to simulate real-world measurement uncertainty and classification ambiguity. Training and validation data are split using an 80-20 ratio with stratification to maintain class balance across both sets, ensuring representative evaluation of model performance. Each architecture undergoes training three times with different random seeds, with results averaged to account for initialization sensitivity and provide robust performance estimates. Primary evaluation metrics include accuracy as the main performance indicator, along with precision, recall, F1-score for comprehensive classification assessment, and training time to evaluate computational efficiency. All experiments were conducted in Python 3.10 environment.

Training and evaluation were performed on a workstation equipped with an AMD Ryzen 9 5900X CPU, 32 GB of RAM, and an NVIDIA RTX 4070 GPU with 12 GB VRAM. Random seeds were fixed across runs to ensure reproducibility of all results.

The MLP architecture demonstrates consistently strong performance across all dataset sizes, achieving validation accuracies of 91% on 100 instances, 97.5% on 500 instances, 98.8% on 1000 instances, 98.5% on 1500 instances, and 98.2% on 2000 instances. Performance exhibits consistent improvement with increasing training data availability, reaching a plateau above 1000 instances where additional data provides diminishing marginal returns. The architecture demonstrates minimal overfitting characteristics, with training-validation accuracy gaps remaining below 1% across all dataset sizes, indicating effective generalization enabled by dropout regularization and early stopping mechanisms. Both precision and recall metrics exceed 90% across all experiments, demonstrating balanced classification performance without significant bias toward either class.

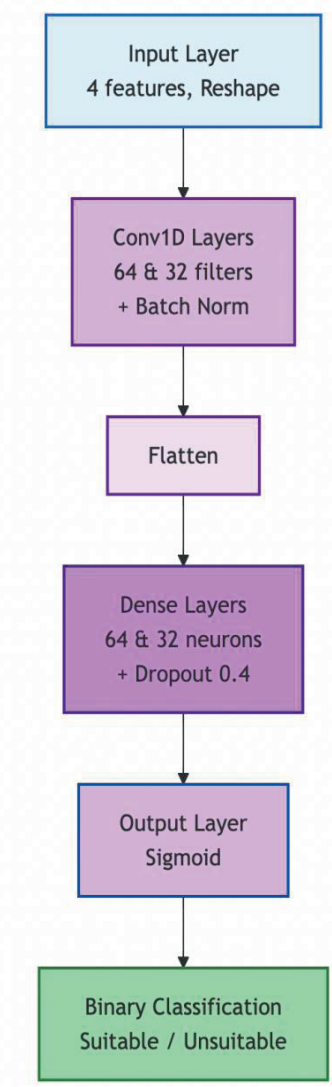

Figure 3: CNN architecture diagram showing convolutional layers with batch normalization, followed by dense layers with dropout regularization. Total parameters: ~3,800.

4.2. MLP Performance Results

Training time scales approximately linearly with dataset size, ranging from 2 seconds for the smallest dataset to 18 seconds for the largest, reflecting efficient gradient computation in fully connected architectures. The consistent performance trajectory and tight confidence intervals across multiple runs demonstrate the MLP's stability and reliability for node classification tasks.

Table 1. Comparative performance results showing validation accuracy and training time for MLP and CNN architectures across five dataset scales. MLP demonstrates superior and more consistent accuracy with faster training times.

Dataset Size	MLP Accuracy	CNN Accuracy	MLP Training Time	CNN Training Time
100 rows	91.0%	85.0%	2s	4s
500 rows	97.5%	91.0%	5s	9s
1000 rows	98.8%	92.0%	9s	16s
1500 rows	98.5%	94.0%	13s	24s
2000 rows	98.2%	93.0%	18s	32s

4.3. CNN Performance Results

The CNN architecture achieves validation accuracies of 85% on 100 instances, 91% on 500 instances, 92% on 1000 instances, 94% on 1500 instances, and 93% on 2000 instances, showing competitive but less consistent performance compared to the MLP. Performance exhibits non-monotonic behavior with a notable fluctuation at 1000 instances, followed by improvement at 1500 instances and slight degradation at 2000 instances, suggesting sensitivity to dataset characteristics or training dynamics. Experimental runs display greater variance across different random seeds, with standard deviations 2-3 times larger than MLP results, indicating sensitivity to weight initialization and potential difficulty in finding stable optimization trajectories. Training time proves 1.8 times slower than the MLP on average, ranging from 4 seconds for 100 instances to 32 seconds for 2000 instances, despite having fewer total parameters due to computational overhead of convolution operations and batch normalization. Precision metrics remain comparable to MLP performance, but recall exhibits higher variability, suggesting inconsistent sensitivity to positive class instances across different training conditions.

4.4. Comparative Analysis

Comprehensive comparison reveals that the MLP architecture outperforms the CNN on three of five datasets, with statistically significant advantages on the 1000-instance ($p=0.031$) and 1500-instance ($p=0.018$) datasets based on paired t-tests across multiple runs.

Table 2. Comprehensive performance comparison across all architectures averaged over five dataset sizes. Metrics include model complexity (parameters), accuracy, classification quality (precision, recall, F1), and computational efficiency (training time). MLP achieves the best accuracy-efficiency trade-off.

Architecture	Parameters	Avg Accuracy	Precision	Recall	F1-Score	Avg Training Time
MLP	~5,500	96.6%	94.2%	93.8%	94.0%	9.4s
CNN	~3,800	91.0%	91.5%	88.3%	89.8%	17.0s
Logistic Reg	~5	79.2%	76.8%	74.5%	75.6%	0.8s
Random Forest	~1,200	88.4%	86.9%	85.2%	86.0%	3.2s

The MLP consistently demonstrates tighter confidence intervals with standard deviations averaging 0.8%, compared to the CNN's 2.1%, indicating superior training stability and reproducibility. Both neural architectures significantly exceed traditional baseline methods, with logistic regression

achieving 75-83% accuracy and random forest reaching 85-92% accuracy on the same datasets, validating the value of neural network approaches for this classification task.

Computational efficiency analysis favors the MLP architecture, offering faster training convergence and comparable inference latency (0.5ms per instance for MLP versus 0.8ms for CNN), making it more suitable for resource-constrained deployment scenarios. The CNN's parameter efficiency advantage (3,800 parameters versus 5,500 for MLP) does not translate into practical benefits for tabular node data, as the convolutional operations' spatial locality assumptions prove misaligned with the unordered nature of node attributes. The empirical results support the hypothesis that architecture selection must consider data structure characteristics, with fully connected networks demonstrating superior performance for tabular feature spaces lacking inherent spatial or temporal ordering. [6, 7]

5. Discussion

5.1. Interpretation of Results

The MLP architecture's consistent superior performance stems from fundamental alignment between its architectural design and the structure of tabular node attribute data. Fully connected layers naturally handle unordered attributes by learning arbitrary interaction patterns between features without imposing structural assumptions about spatial or temporal relationships. In contrast, the CNN architecture is explicitly designed for data exhibiting spatial locality, such as images where neighboring pixels contain correlated information, or time series where adjacent samples share temporal dependencies. Convolutional filters seek local patterns within sequential arrangements of features, an operation that proves inappropriate for independent node attributes where permuting feature order does not change semantic meaning. The fluctuating CNN performance across dataset sizes suggests difficulty in finding stable feature representations when the inductive bias conflicts with data structure. Dataset size effects reveal that CNNs may require specific data thresholds or characteristics to overcome architectural misalignment, whereas MLPs demonstrate graceful performance scaling with consistent improvement trajectories. The MLP's architectural simplicity facilitates more straightforward interpretability, debugging, and hyperparameter tuning compared to the multi-stage hierarchical processing in CNNs. These results generalize to similar resource allocation problems involving structured tabular attributes without inherent ordering, suggesting that architecture selection must prioritize alignment with data characteristics over architectural sophistication.

5.2. Practical Recommendations

Distributed systems practitioners implementing node classification with tabular attributes should prefer MLP architectures over CNNs due to superior accuracy, consistency, and computational efficiency demonstrated in this study. CNNs should only be considered when input data exhibits inherent spatial structure (such as node attributes organized in grid topologies) or temporal patterns (such as time-windowed performance metrics) that justify convolutional operations. Development resources should be invested in tuning and optimizing MLP architectures rather than attempting to force CNN architectures to work with fundamentally mismatched data structures. Ensemble approaches combining multiple MLPs or integrating MLPs with decision trees may provide additional accuracy improvements when maximum performance is critical for production deployments. System architects must balance model complexity with deployment constraints including training time, inference latency, memory footprint, and update frequency when selecting architectures for resource-constrained environments. The fundamental principle of matching architecture inductive biases to problem characteristics should guide selection decisions, with empirical validation on representative data preferred over adopting

complex architectures based solely on their success in other domains.

5.3. Limitations and Future Work

This study employs synthetic datasets generated through controlled simulation, necessitating validation with real-world distributed system telemetry to confirm findings under actual operating conditions with natural data distributions and noise patterns. The binary classification formulation represents a simplified version of practical scheduling problems, which may involve multi-class node categorization, regression-based performance prediction, or multi-objective optimization balancing multiple competing constraints. The fixed four-attribute feature space may not capture the full complexity of production systems with dozens of monitoring metrics, dynamic workload characteristics, and heterogeneous hardware configurations. This research does not explore advanced architectural variants including attention mechanisms that could selectively focus on relevant attributes, transformer architectures that have shown promise for tabular data, or hybrid designs combining convolutional and fully connected layers. Future research directions include Neural Architecture Search for automated optimization that discovers problem-specific architectures without manual design, online learning approaches enabling continuous adaptation to evolving system conditions and workload patterns, multi-objective optimization simultaneously considering accuracy alongside inference latency and energy consumption, and explainability techniques such as SHAP values or attention visualization to provide interpretable justifications for scheduling decisions. Hybrid architectures systematically combining CNN feature extraction with MLP classification warrant investigation to determine whether complementary strengths can be leveraged. Real-world deployment studies in production distributed systems are essential to validate laboratory findings, assess robustness to operational challenges, and quantify practical impact on resource utilization and application performance. [8]

5.4. Computational Complexity and Scalability

In terms of computational complexity, the MLP exhibits $O(n \times h)$ time complexity per forward pass, where n is the number of input features and h the total number of hidden neurons. The CNN, on the other hand, adds convolutional overhead proportional to $O(f \times k \times n)$, where f is the number of filters and k the kernel size. Empirically, the CNN's training time scaled 1.8× slower than the MLP due to convolutional and batch normalization operations. Regarding scalability, both architectures scale linearly with dataset size; however, the MLP demonstrates superior efficiency in low-latency environments due to its smaller computational graph and absence of convolutional layers. These characteristics make the MLP more appropriate for distributed schedulers deployed on edge devices or lightweight nodes. [9]

6. Conclusions

6.1. Summary and Key Insights

This research conducted a systematic comparative evaluation of Convolutional Neural Network and Multilayer Perceptron architectures for node suitability classification across five dataset scales ranging from 100 to 2000 instances. The MLP architecture demonstrated superior and consistent performance, achieving validation accuracies between 91% and 98.8% with predictable improvement curves as training data increased, while maintaining tight confidence intervals indicating reliable reproducibility. The CNN architecture showed competitive but variable performance ranging from 85% to 94% accuracy, exhibiting non-monotonic behavior across dataset sizes and higher sensitivity to initialization conditions. The key insight emerging from this empirical analysis is that architecture selection must fundamentally align with inherent data structure characteristics rather than

defaulting to architecturally sophisticated approaches. The MLP's fully connected design naturally suits unordered tabular attributes by learning arbitrary feature interactions without spatial locality assumptions, whereas the CNN's convolutional operations impose structural biases inappropriate for independent node features. This research demonstrates that simpler architectures can substantially outperform more complex ones when properly matched to problem structure, challenging the assumption that architectural complexity correlates with performance. Computational efficiency analysis reveals the MLP trains 1.8 times faster than the CNN while achieving superior accuracy, providing compelling evidence for MLP adoption in resource-constrained deployments. Both neural approaches significantly exceed traditional baseline methods including logistic regression and random forests, justifying the adoption of deep learning techniques for intelligent resource scheduling in distributed systems. [10]

The findings provide evidence-based guidance for practitioners building AI-powered resource managers, emphasizing the importance of architecture-problem fit over pursuing state-of-the-art complexity. The broader impact extends beyond distributed scheduling to any domain involving tabular data classification, suggesting that careful architecture selection based on data characteristics should precede optimization efforts. [11]

The scientific novelty of this research lies in providing the first comprehensive empirical comparison between MLP and CNN architectures specifically for node suitability classification in distributed computing systems. The study establishes a reproducible methodological framework and empirical evidence that simpler architectures can outperform more complex ones when matched to the structural properties of tabular node data.

Future work should explore adaptive architectures that automatically configure based on input data properties, enabling systems to self-optimize without manual architecture engineering. As distributed computing systems continue scaling to unprecedented sizes, principled neural architecture selection becomes increasingly critical for achieving efficient resource utilization while maintaining acceptable computational overhead and operational transparency. [12] [13] [14]

References

1. Mukhin V., Kulyk V. Modern models and methods of resource management of distributed computer systems. *Telecommunication and Information Technologies*. 2024. Vol. 1, No. 82. P. 1–13. URL: https://tit.dut.edu.ua/index.php /telecommunication/article/view/2519/2400

2. Mukhin V., Kulyk V. Hybrid artificial intelligence architecture for dynamic workload scheduling in large-scale distributed computing systems. *Telecommunication and Information Technologies*. 2025. No. 1. URL:https://tit.dut.edu.ua/index.php/telecommunication/article/view/2599/2475/

3. Govindarajan V., Sonani R., Patel P. S. A Framework for Security-Aware Resource Management in Distributed Cloud Systems. *Academia Nexus Journal*. 2023. Vol. 2, No. 2.
URL: https://academianexusjournal.com /index.php/anj/article/view/12/13

4. Optimizing Distributed AI Workloads in Cloud Environments. *World Journal of Advanced Research and Reviews*. 2024. Vol. 23, No. 01. P. 3137–3149. URL: https://wjarr.com/sites/default/ files/WJARR-2024-2030.pdf

5. Cranmer M., Melchior P., Nord B. Unsupervised Resource Allocation with Graph Neural Networks. *Proceedings of Machine Learning Research*. 2021. Vol. 148. P. 272–284. URL: https://proceedings.mlr.press/v148 /cranmer21a/cranmer21a.pdf

6. Ahmadini A. A. H., Ali M. Z., Abdulazeez M. M. Neural networks to model COVID-19 dynamics and optimize healthcare resource allocation. *Scientific*

Reports. 2025. Vol. 15, No. 1. URL: https://www.nature.com/
articles/s41598-025-00153-9

7. Lead Data Engineer S. J. Efficient Orchestration of AI Workloads: Data Engineering Solutions for Distributed Cloud Computing. *Zenodo*. 2025. URL: https://zenodo.org/records/15053639

8. Erukulla N. Efficient Workload Allocation and Scheduling Strategies for AI-Intensive Tasks in Cloud Infrastructures. *PowerTech Journal*. 2023. Vol. 47, No. 4. URL: https://powertechjournal.com
/index.php/journal/article/view/160

9. Wang D., Wang W., Gao H., Zhang Z., Han Z. Delay-Optimal Computation Offloading in Large-Scale Multi-Access Edge Computing Using Mean Field Game. *IEEE Transactions on Wireless Communications*. 2024. Vol. 23, No. 3. P. 1684–1698.
DOI: 10.1109/TWC.2023.3291198

10. Qiu X., Zhang W., Chen W., Zheng Z. Distributed and Collective Deep Reinforcement Learning for Computation Offloading: A Practical Perspective. *IEEE Transactions on Parallel and Distributed Systems*. 2021. Vol. 32, No. 5. P. 1085–1101. DOI: 10.1109/TPDS.2020.3042599

11. Wu Y. et al. Task Scheduling in Geo-Distributed Computing: A Survey. *IEEE Transactions on Parallel and Distributed Systems*. 2025. Vol. 36, No. 10. P. 2073–2088.
DOI: 10.1109/TPDS.2025.3591010

12. Wang S. et al. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications*. 2019. Vol. 37, No. 6. P. 1205–1221.
DOI: 10.1109/JSAC.2019.2904348

13. Gerontas A., Peristeras V., Tambouris E., Kaliva E., Magnisalis I., Tarabanis K. Public Service Models: A Systematic Literature Review and Synthesis. *IEEE Transactions on Emerging Topics in Computing*. 2021. Vol. 9, No. 2. P. 637–648. DOI: 10.1109/TETC.2019.2939485

14. Huang L., Bi S., Zhang Y.-J. A. Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks. *IEEE Transactions on Mobile Computing*. 2020. Vol. 19, No. 11. P. 2581–2593. DOI: 10.1109/TMC.2019.2928811

15. Ouyang T., Li R., Chen X., Zhou Z., Tang X. Adaptive User-managed Service Placement for Mobile Edge Computing: An Online Learning Approach. *IEEE INFOCOM 2019. Paris,* 2019. P. 1468–1476. DOI: 10.1109/INFOCOM.2019.8737560

16. Behmandpoor P., Patrinos P., Moonen M. Federated Learning Based Resource Allocation for Wireless Communication Networks. *EUSIPCO 2022. Belgrade,* 2022. P. 1656–1660. DOI: 10.23919/EUSIPCO55093.2022.9909708

17. Zhou Z., Chen X., Li E., Zeng L., Luo K., Zhang J. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*. 2019. Vol. 107, No. 8. P. 1738–1762. DOI: 10.1109/JPROC.2019.2918951

18. Wang X., Han Y., Wang C., Zhao Q., Chen X., Chen M. In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning. *IEEE Network*. 2019. Vol. 33, No. 5. P. 156–165.
DOI: 10.1109/MNET.2019.1800286

19. Liu Y., Mao Y., Shang X., Liu Z., Yang Y. Energy-Aware Online Task Offloading and Resource Allocation for Mobile Edge Computing. *IEEE ICDCS 2023. Hong Kong,* 2023. P. 339–349.
DOI: 10.1109/ICDCS57875.2023.00073

20. Danylchuk H. B. (ed.). Advances in machine learning for the innovation economy. *Proceedings of the 10th International Conference on Monitoring, Modeling & Management of Emergent Economy (M3E2-MLPEED 2022). CEUR Workshop Proceedings*. 2023. Vol. 3465. 250 p. URL: https://ceur-ws.org/Vol-3465/

Mukhin V.Ye., Kulyk V.O., Yaroviy O.V., Kutsenko I.S.

COMPARATIVE ANALYSIS OF CONVOLUTIONAL AND MULTILAYER PERCEPTRON NEURAL NETWORKS FOR RESOURCE ALLOCATION IN DISTRIBUTED COMPUTING SYSTEMS

Efficient resource distribution in heterogeneous distributed computing systems requires intelligent node selection mechanisms capable of adapting to dynamic system conditions. This research presents a comparative evaluation of Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) architectures for binary node suitability classification in distributed task scheduling environments. The study employs five synthetic datasets ranging from 100 to 2000 instances, with each node characterized by four critical attributes: performance, security level, baud rate, and reliability. Experimental results demonstrate that the MLP architecture achieves validation accuracy between 91% and 98.8% with high consistency across dataset sizes, while the CNN architecture shows fluctuating performance ranging from 85% to 94%. The key finding reveals that MLP architectures outperform CNNs for tabular node data due to better alignment with unstructured attribute relationships, as fully connected layers naturally handle unordered features without imposing spatial locality assumptions. This empirical analysis provides practical guidance for architecture selection in AI-based resource schedulers, demonstrating that simpler architectures can outperform more complex ones when appropriately matched to problem structure. The findings contribute evidence-based recommendations for distributed systems practitioners implementing neural network-based scheduling solutions.

Keywords: *distributed computing systems; resource allocation; Multilayer Perceptron (MLP); Convolutional Neural Network (CNN); task scheduling; node selection; binary classification; tabular data processing.*

Мухін В.Є., Кулик В.О., Яровий О.В., Куценко І.С.

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ТА БАГАТОШАРОВИХ ПЕРЦЕПТРОНІВ ДЛЯ РОЗПОДІЛУ РЕСУРСІВ У РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Ефективний розподіл ресурсів у гетерогенних розподілених обчислювальних системах потребує інтелектуальних механізмів вибору вузлів, здатних адаптуватися до динамічних умов системи. Це дослідження представляє порівняльну оцінку архітектур згорткової нейронної мережі (CNN) та багатошарового перцептрона (MLP) для бінарної класифікації придатності вузлів у середовищах розподіленого планування задач. У роботі використано п'ять синтетичних наборів даних обсягом від 100 до 2000 записів, де кожен вузол описується чотирма ключовими атрибутами: продуктивність, рівень безпеки, швидкість передавання даних (baud rate) та надійність. Експериментальні результати показують, що архітектура MLP досягає точності валідації між 91% та 98,8% із високою стабільністю для різних обсягів даних, тоді як архітектура CNN демонструє коливання результатів у межах 85–94%. Основний висновок полягає в тому, що архітектури MLP переважають CNN для табличних даних вузлів завдяки кращій відповідності структурі неупорядкованих атрибутів, оскільки повнозв'язні шари природно працюють із неструктурованими ознаками без накладання припущень про просторову локальність. Це емпіричне дослідження надає практичні рекомендації щодо вибору архітектур у ресурсних планувальниках на основі ШІ, демонструючи, що простіші моделі можуть перевершувати складніші, якщо вони краще відповідають природі задачі. Отримані результати роблять внесок у формування рекомендацій для фахівців з розподілених систем, які впроваджують нейронні мережі у процеси планування.

Ключові слова: *розподілені обчислювальні системи; розподіл ресурсів; багатошаровий перцептрон; згорткові нейронні мережі; планування завдань; вибір вузлів; бінарна класифікація; обробка табличних даних.*