Learning-Driven Task Orchestration for Latency-Aware Scheduling in Edge–Fog Systems

E. Mahesh[1], K. Srinivas[2*], Jakkam Anusha[3], Panniru Dhanusha[3], Velan Bhaskar[3], Md Khaja Siraj Uddin[3]

[1]Assistant Professor, [2]Associate Professor & Head, [3]UG Student, [1,2,3]Department of Computer Science and Engineering (Data science)

[1,2,3]Vaagdevi Engineering College, Bollikunta, Warangal, 506005, Telangana, India

*Correspondence: K. Srinivas (srinivas_k@vecw.edu.in)

Abstract

Edge–fog computing environments are increasingly vital for handling large-scale, real-time workloads generated by digital services, Internet of Things (IoT), and distributed applications. However, efficient task scheduling remains a major challenge due to dynamic resource availability, heterogeneous architectures, and diverse workload patterns. Traditional approaches, such as static heuristics and rule-based schedulers, as well as conventional machine learning (ML) models, often fail to adapt to changing system conditions. These methods frequently misclassify task requirements, lack the ability to predict multiple scheduling parameters simultaneously, and struggle to scale effectively, leading to reduced system throughput and inefficient resource utilization. To address these limitations, this study proposes an adaptive, data-driven scheduling framework that combines preprocessing, exploratory data analysis (EDA), and multi-model prediction. The proposed Convolutional Recurrent Network with Greedy Rule Interpretable Machine (CRN-GRIM) captures temporal workload patterns while maintaining interpretability through rule-based reasoning. Additional models, including Passive Aggressive Classifier (PAC), Gaussian Naive Bayes (GNB), and K-Nearest Neighbors Classifier (KNNC), are incorporated for comparative evaluation and robustness. The framework predicts multiple scheduling targets are job priority, scheduler type, and resource allocation within a unified multi-output model, enhancing decision accuracy. A lightweight storage system and web-based interface enable real-time prediction, visualization, monitoring, and retraining. This approach delivers a scalable, intelligent, and efficient scheduling solution for modern edge–fog computing environments.

Keywords: Edge–Fog Computing, Task Scheduling, Real-Time Workloads, Resource Allocation, Multi-Output Prediction, Convolutional Recurrent Network, CRN-GRIM, Greedy Rule-Based Learning.

1. Introduction

Today, the world has seen a growth in mobile devices and computers' everyday usage by individuals and organizations. Applications and sensors of electronic devices are used to produce data, usually a massive amount of it. As a result, many companies must take responsibility for routinely maintaining vast volumes of data. Companies currently need a dynamic information management architecture because of the transition to cloud computing and the benefits of this shift, such as scalability, availability, and pay-as-you-use features [1]. Cloud computing has made many services available that include platform, software, and infrastructure as services heading toward anything can be presented as service, as shown in figure 1. Still, it is not always feasible to transfer big data generated by sensors to the cloud for processing and storage. Moreover, several IoT applications require faster processing, only present-day cloud incapable of meeting such applications' latency requirement [2]. The problem is approached using FC, which involves harnessing the computing power of devices close to a user to help with the storage and processing of data [3]. FC's different goals include efficiency improvements, data

size reduction required to be transmitted to the cloud for data processing, analysis, and storage. The leading cause for this is often performance, but security and compliance can be other reasons [4]. Recently the AI algorithms have been applied to IoT data analysis. End-user devices at the lowest layer of the network carry many unwanted features, such as insufficient memory, an inadequate low communication bandwidth, low processing power, and heterogeneous hardware that differ from the cloud infrastructure [5].

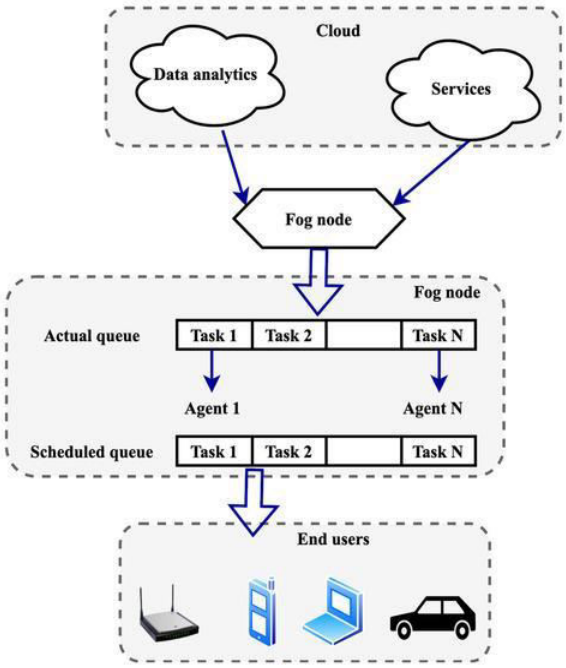

Figure. 1: Intelligent task scheduling in edge-fog computing environments

Computing technologies have advanced in various areas throughout the last decade, such as AI, GPU computing, cloud computing, and various hardware enhancements [6]. ML is the most widely implemented AI algorithm in various fields. Researchers have used ML to overcome networking problems in several earlier studies, including network routing, security, traffic engineering, and resource allocation [7]. ML plays a significant role in creating a smart/intelligent environment where autonomous management and operation are concerned. The significance of ML is extended to IoT, as without it, IoT will not be possible in terms of performing functional, monitoring, or preprocessing tasks [8]. Moreover, meeting the IoT's diverse QoS demands remains a formidable issue due to the IoT devices' resources limited nature. Thus, describing ML concerning fog, cloud, and edge computing for IoT deployment is critical [9].

2. LITERATURE SURVEY

Leena, et al. [10] proposed work deals with an efficient algorithm of task scheduling amidst the fog nodes with an improvement in the energy utility. The research provided an improved hybrid heuristics approach that helps in the proper utilization of the fog nodes that are limited in terms of computational resources and energy. It compares the proposed work with the existing heuristic algorithms and is proved to be superior, and a mathematical analysis of the objectives has also been presented. The proposed algorithm shows a reduction in service time by 18% as compared to neural networks and improvement in energy efficiency of about 29% as compared to the existing heuristics with a significant 40% reduction in the average delay of the system as compared to the use of Deep neural networks.

Tian, et al. [11] adopted this method is based on the Ant Lion Optimizer (ALO. Firstly, chaotic mapping is to initialize the population, and the quality of the initial population is improved; secondly, the

Adaptive Random Wandering (ARW) method is designed to improve the solution efficiency; finally, the improved Dynamic Opposite Learning Crossover (DOLC) strategy is embedded in the generation-hopping stage of the ALO to enrich the diversity of the population and improve the optimization-seeking ability of ALO. HALO is used to optimize the scheduling scheme of fog computing tasks. The simulation experiments are conducted under different data task volumes, compared with several other task scheduling algorithms such as the original algorithm of ALO, Genetic Algorithm (GA), Whale Optimizer Algorithm (WOA) and Salp Swarm Algorithm (SSA). HALO has good initial population quality, fast convergence speed, and high optimization-seeking accuracy. The scheduling scheme obtained by the proposed method in this paper can effectively reduce the latency of the system and reduce the energy consumption of the system. Anand, et al. [12] proposed the scheduling plays a crucial role in offloading decisions in multi-access edge computing. The motivations for scheduling are to improve the quality of the experience, reduce latency, and increase performance. They explore the various scheduling techniques available for MEC systems, including static scheduling, dynamic scheduling, heuristics, meta-heuristics and hybrid scheduling. They analysed the advantages and disadvantages of each technique and discuss how they can be used to optimize the performance of MEC applications. They also present a case study of an MEC system and demonstrate how the various scheduling techniques can be used to maximize its performance.

Ahanger, et al. [13] presented the IoT-Edge-Fog Computing a trio-logical model for decentralized computing in a time-sensitive manner. To address the rising need for real-time information processing and decision modelling, task allocation among dispersed Edge Computing nodes has been a major challenge. State-of-the-art task allocation techniques such as Min–Max, Minimum Completion time, and Round Robin perform task allocation, but several limitations persist including large energy consumption, delay, and error rate. Henceforth, the current work provides a Quantum Computing-inspired optimization technique for efficient task allocation in an Edge Computing environment for real-time IoT applications. Furthermore, the QC-Neural Network Model is employed for predicting optimal computing nodes for delivering real-time services. To acquire the performance enhancement, simulations were performed by employing 6, 10, 14, and 20 Edge nodes at different times to schedule more than 600 heterogeneous tasks. Empirical results show that an average improvement of 5.02% was registered for prediction efficiency. Similarly, the error reduction of 2.03% was acquired in comparison to state-of-the-art techniques. Abdulkareem, et al. [14] reviewed AI applications within fog computing environments, adhering to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) methodology to ensure rigorous analysis. The review identifies critical issues such as resource constraints, transparency in AI-driven security systems, and the need for adaptable AI models to address evolving security threats. In response, innovative solutions such as lightweight AI models (e.g., Pruned Neural Networks, Quantized Models, Knowledge Distillation), Explainable AI (XAI) (e.g., Model-Agnostic Methods, Feature Importance Analysis, Rule-Based Approaches), and federated learning are proposed. Additionally, a novel taxonomy is introduced, categorizing AI techniques into resource management, security enhancement, and privacy preserving methods, offering a structured framework for researchers and practitioners.

Sun, et al. [15] proposed a hierarchical data job scheduling strategy Based on Intelligent Sensor-Cloud in Fog Computer (HDJS). HDJS dynamically adjusts the priority of the job to avoid job starvation and maximize the use of resources, uses the key frame to the resource occupied information, distributes the frame sequence to the unit, and then combines the intra frame distribution strategy to balance the load between the nodes. The experimental results show the proposed strategy may be possible to avoid the operation of hunger and resource fragmentation problems, make full use of the advantages of multi-core and multi-thread, improve system resource utilization, and shorten the execution time and response time.

3. Proposed System

The proposed methodology establishes a systematic approach for building an intelligent, data-driven task scheduling system capable of operating efficiently under dynamic and resource-constrained conditions. It incorporates a structured pipeline that begins with data ingestion, preprocessing, and feature extraction, followed by multi-model learning and predictive analysis. Classical machine learning models are combined with a hybrid deep-learning and rule-based classifier to improve accuracy, adaptability, and decision reliability described in figure 2. A lightweight storage mechanism ensures smooth data management while a web-based interface facilitates real-time interaction, visualization, and monitoring. Continuous model retraining further enhances performance, enabling the methodology to adapt to evolving workload behaviours and maintain long-term efficiency.

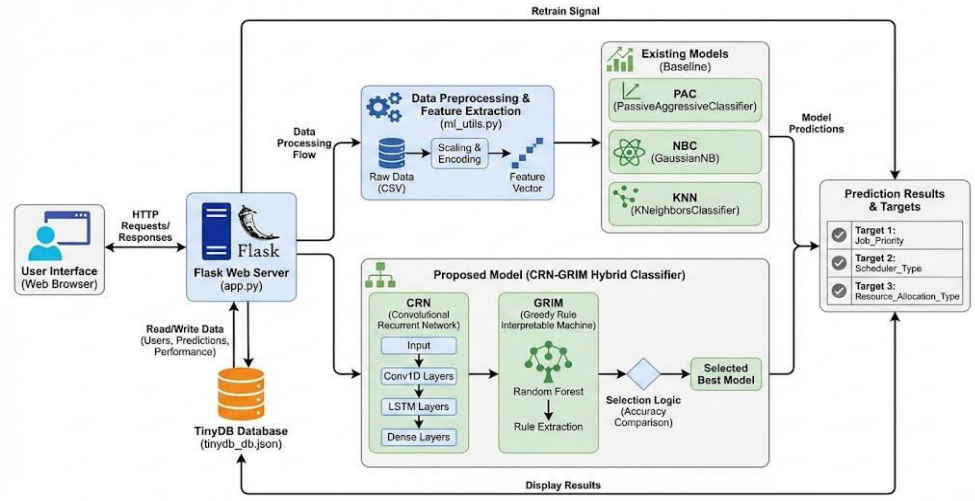

Figure. 2: System architecture

User Interface (Web Browser)

- The user interacts with the system through a browser-based graphical interface.
- They can perform actions like login, dataset upload, single/batch prediction, EDA viewing, and model retraining.
- All user interactions are converted into HTTP requests and sent to the Flask web server.

Flask Web Server (app.py)

- The Flask backend receives the requests from the UI and routes them to the correct functions.
- It controls user authentication, prediction requests, data retrieval, and model management.
- The server reads from and writes to the TinyDB database, ensuring seamless communication between UI, models, and stored records.
- It also triggers model retraining signals when the admin initiates re-training.

TinyDB Database (tinydb_db.json)

- TinyDB stores all persistent information needed for system operation.

- It keeps records of registered users, prediction history, model performance logs, and retraining metadata.
- The database interacts with the Flask server for fast read/write operations.
- Its JSON-based structure ensures portability for edge–fog computing environments.

Raw Data (CSV Input)

- The workload dataset (CSV file) serves as the primary input source.
- It contains numerical and categorical features such as CPU utilization, error rate, throughput, memory usage, bandwidth, and active users.
- This data flows directly into the preprocessing pipeline for feature extraction.

Data Preprocessing & Feature Extraction (ml_utils.py)

- The raw dataset undergoes cleaning, scaling, and encoding to prepare it for ML model input.
- Numeric features are standardized, while categorical features are label-encoded.
- Feature vectors are generated for each task instance, creating a consistent numerical format for training and prediction.
- This processed feature set is forwarded to all baseline and hybrid models.

Existing Baseline Models (PAC, GNB, KNNC)

- The pre-processed feature vector is passed to the baseline models:
  - **PAC:** Fast large-scale linear model.
  - **GNB:** Probabilistic classifier for feature distributions.
  - **KNNC:** Distance-based instance classifier.
- These models generate independent predictions for each of the three target variables:
  1. Job Priority
  2. Scheduler Type
  3. Resource Allocation Type
- Their results contribute to overall model comparison and evaluation.

Proposed Hybrid CRN-GRIM Classifier: This is the core intelligent fusion model of the system.

1. **CRN**
   - The CRN processes the feature vector through:
     - Convolutional layers to learn feature interactions
     - Long Short-Term Memory (LSTM) layers to capture sequence and temporal dependencies
     - Dense layers to generate final prediction logits
   - This deep learning path produces one set of predictions.
2. **GRIM**
   - GRIM uses a compressed Random Forest (RF) to generate predictions.

- It extracts rule-based decision structures for interpretability.
- Produces a second set of predictions.

Selection Logic (Best Model Selection)

- The framework evaluates CRN and GRIM output accuracies.
- The model with the highest accuracy becomes the Selected Best Model for final prediction.
- This "deep-decision fusion" ensures maximum accuracy while retaining interpretability.

Prediction Results & Target Output

- After predictions from the baseline and hybrid models, the system outputs results for the three main scheduling tasks:
  - Target 1: Job Priority
  - Target 2: Scheduler Type
  - Target 3: Resource Allocation Type
- The results are displayed in the UI and stored in TinyDB.
- The system may also compute confidence scores, probabilities, and model-wise comparisons.

Retrain Signal (Admin-Initiated)

- When the admin chooses to retrain models from the UI, a retrain signal is sent to the pipeline.
- The preprocessing module reloads updated datasets.
- Baseline models and the CRNGRIM hybrid classifier are retrained to improve accuracy.
- Updated model performance stats are stored in TinyDB, completing the adaptive learning cycle.

4. Results description

The results of the proposed system demonstrate the effectiveness of integrating multiple machine learning models with a hybrid CRN-GRIM approach for workload classification in edge–fog environments. The system was evaluated using key performance metrics such as accuracy, precision, recall, F1-score, and AUC to ensure comprehensive assessment. Experimental outcomes indicate that the hybrid CRN-GRIM model achieved superior performance compared to traditional classifiers like PAC, GNB, and KNN. The inclusion of deep learning through LSTM layers enabled better pattern recognition, while the GRIM component enhanced interpretability. Visualization tools such as confusion matrices and ROC curves further validated the reliability of the predictions. Additionally, the system maintained consistent performance under varying workload conditions, demonstrating robustness and scalability.

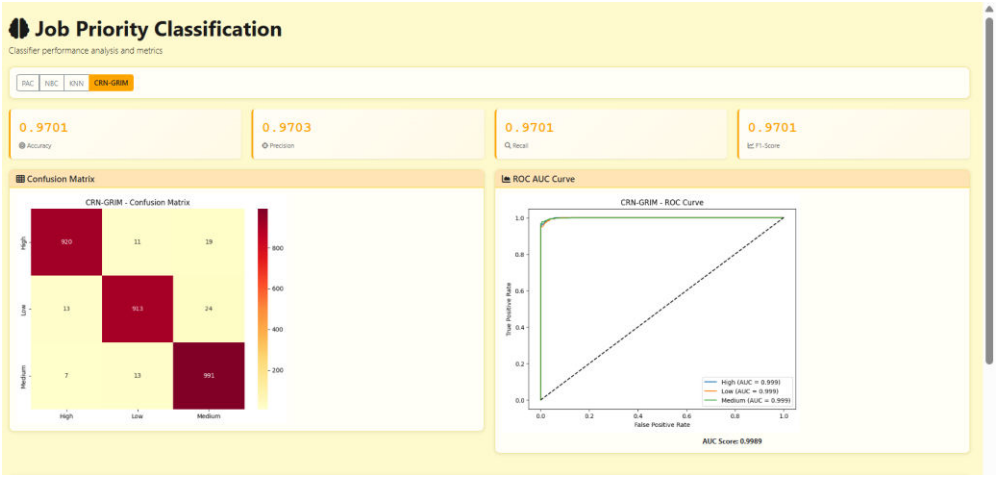

Figure. 3: Detailed classification performance analysis for job priority (CRN-GRIM Classifier)

Figure 3 displays the performance of the hybrid CRN-GRIM classifier on the Job Priority prediction task. It shows the key metrics accuracy, precision, recall, and F1-score at the top. Below, a confusion matrix reveals the classification results across all priority classes, and a multi-class ROC curve with individual AUC values per class highlights the model's strong separation capability. Classifier tabs allow instant switching among models for direct comparison on the Job Priority target.

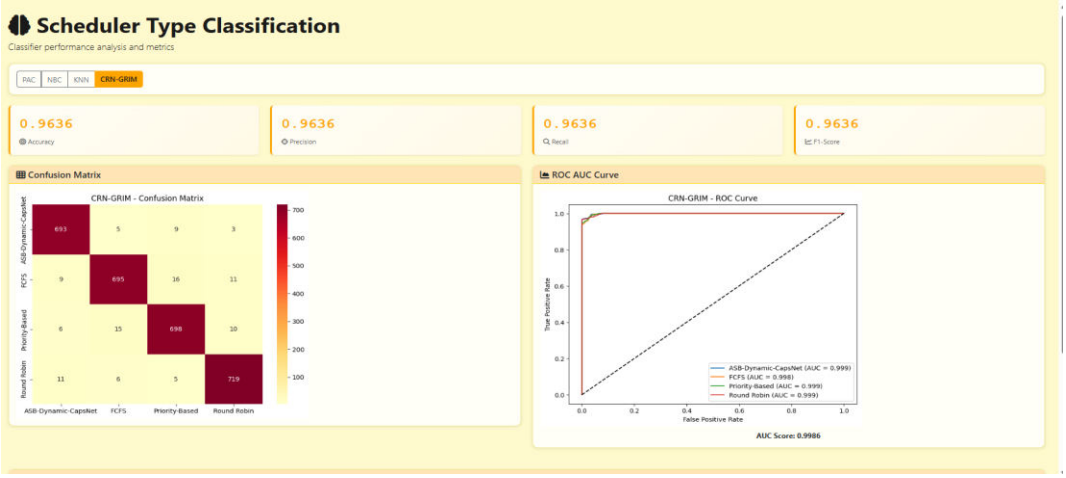

Figure. 4: Detailed classification performance analysis for scheduler type (CRN-GRIM Classifier)

Figure 4 showcases the performance of the hybrid CRN-GRIM classifier on the Scheduler Type prediction task. It displays near-perfect accuracy, precision, recall, and F1-score at the top. A confusion matrix clearly demonstrates highly accurate classification across all scheduler categories with minimal misclassifications, while the multi-class ROC curve with extremely high AUC values for each scheduler type confirms outstanding discriminative power. Classifier selection tabs allow instant comparison with other models on the same target.

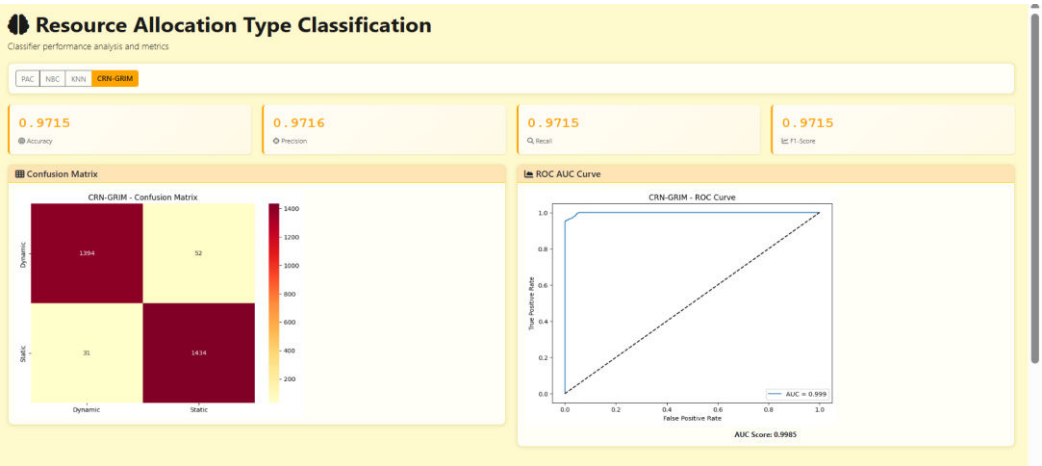

Figure. 5: Detailed classification performance analysis for resource allocation type (CRN-GRIM Classifier)

Figure 5 demonstrates the performance of the hybrid CRN-GRIM classifier on the Resource Allocation Type prediction task. It exhibits near-perfect accuracy, precision, recall, and F1-score. The confusion matrix shows extremely high correct classification rates with only minimal misclassifications between dynamic and static allocation strategies. The accompanying ROC curve, with an AUC very close to 1.0, confirms exceptional discriminative capability for distinguishing between the two resource allocation types. Classifier selection tabs enable immediate comparison with other models on the same Resource Allocation Type target.

Figure 6 displays the outcome of a batch prediction operation performed on an uploaded dataset. It presents a tabular view listing the first set of processed rows, with each row showing the individual predictions generated by all four classifiers (PAC, GNB, KNNC, and CRN-GRIM) for the three scheduling decisions: Job Priority, Scheduler Type, and Resource Allocation Type. The table enables users to quickly compare the predictions of different models' side-by-side for multiple workload records simultaneously, facilitating bulk analysis and decision-making in real-world deployment scenarios.

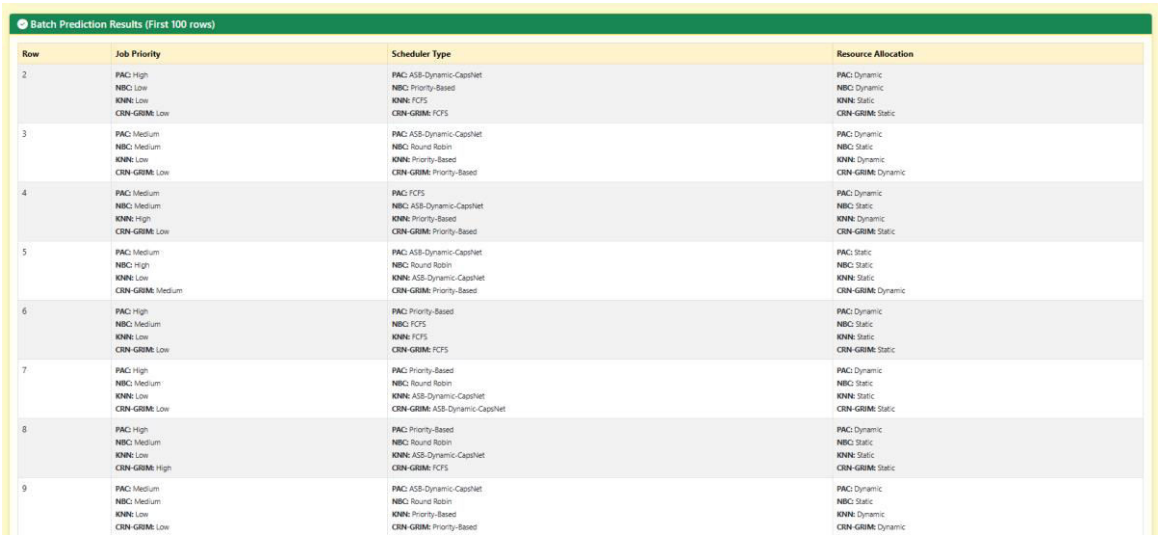

Figure 6: Batch prediction results interface

Table. 1: Performance comparison of classifiers for job priority prediction

Classifier	Accuracy	Precision	Recall	F1-Score

PAC	0.3432	0.3441	0.3432	0.3145
GNB	0.3507	0.3485	0.3507	0.3314
KNNC	0.4792	0.4817	0.4792	0.4796
CRN-GRIM	0.9701	0.9703	0.9701	0.9701

Table. 2: Performance comparison of classifiers for scheduler type prediction

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**
PAC	0.2590	0.2587	0.2590	0.2559
GNB	0.2707	0.2694	0.2707	0.2662
KNNC	0.3957	0.3975	0.3957	0.3938
CRN-GRIM	0.9636	0.9636	0.9636	0.9636

Table 1 presents a comparative evaluation of classifiers for Job Priority prediction. Traditional models such as PAC, GNB, and KNNC deliver moderate performance, with KNNC achieving the highest accuracy among them at 0.4792. Although these baseline models capture certain workload trends, their relatively lower precision, recall, and F1-scores indicate limited ability to generalize across complex job patterns. In contrast, the CRN-GRIM hybrid classifier significantly outperforms all others, achieving an exceptional accuracy of 0.9701, demonstrating its strong capability in learning temporal features and extracting interpretable rules. This highlights the superiority of the hybrid deep-decision fusion model for priority-based scheduling tasks.

Table 2 shows the performance comparison of different classifiers for predicting Scheduler Type. The PAC and GNB models show low predictive capability, with accuracy values around 0.26–0.27, indicating difficulty in handling diverse scheduling categories. The KNNC model performs better with an accuracy of 0.3957, showing its effectiveness in capturing instance-based similarities. However, the CRN-GRIM classifier again leads with an outstanding accuracy of 0.9636, along with equally strong precision, recall, and F1-score. This demonstrates that the hybrid model successfully captures deeper structural and temporal patterns required for selecting appropriate scheduling strategies.

Table. 3: Performance comparison of Classifiers for Resource Allocation Type Prediction

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**
PAC	0.5040	0.5227	0.5040	0.3985
GNB	0.5170	0.5169	0.5170	0.5142
KNNC	0.6245	0.6245	0.6245	0.6245
CRN-GRIM	0.9715	0.9716	0.9715	0.9715

Table 3 highlights the comparative performance of classifiers for Resource Allocation Type prediction. Among the baseline models, KNNC achieves the highest accuracy of 0.6245, indicating reasonable capability in identifying resource distribution patterns, while PAC and GNB show moderate to low performance. Despite this, the CRN-GRIM model far surpasses all others with an accuracy of 0.9715

and consistently strong evaluation metrics across all four categories. The results confirm that the hybrid model effectively integrates deep feature extraction with rule-based interpretability, enabling highly reliable predictions for resource allocation decisions.

5. Conclusion

The proposed Deep-Decision Fusion Framework successfully integrates classical machine learning models with the hybrid CRN-GRIM classifier to deliver highly accurate predictions across all three scheduling tasks Job Priority, Scheduler Type, and Resource Allocation Type. The system demonstrates clear performance improvements, where CRN-GRIM consistently outperforms PAC, GNB, and KNNC with accuracies exceeding 96–97%, compared to the baseline models which remain within the 25–62% range. This significant improvement results from combining deep temporal feature extraction with interpretable rule-based decision mechanisms, allowing the system to model complex workload behaviours effectively. The web-based implementation ensures seamless data preprocessing, visualization, model training, and prediction, while TinyDB provides lightweight and persistent storage of user and prediction records. The comparative analysis validates the effectiveness of the hybrid approach, proving that deep-decision fusion achieves superior generalization, robustness, and reliability for dynamic task scheduling in edge–fog environments.

References

[1] J. C. Guevara, R. da S. Torres, and N. L. S. da Fonseca, "On the classification of fog computing applications: A machine learning perspective," J. Netw. Comput. Appl., vol. 159, p. 102596, Jun. 2020, https://doi.org/10.1016/j.jnca.2020.102596

[2] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, and T. Westerlund, "Edge and Fog Computing Enabled AI for IoT-An Overview," in 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Mar. 2019, pp. 51–56, https://doi.org/10.1109/AICAS.2019.8771621

[3] L. M. Herger, M. Bodarky, and C. Fonseca, "Breaking Down the Barriers for Moving an Enterprise to Cloud," in 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), Jul. 2018, vol. 2018-July, pp. 572–576, https://doi.org/10.1109/CLOUD.2018.00079

[4] A. Rashid and A. Chaturvedi, "Cloud Computing Characteristics and Services A Brief Review," Int. J. Comput. Sci. Eng., vol. 7, no. 2, pp. 421–426, Feb. 2019, https://doi.org/10.26438/ijcse/v7i2.421426

[5] M. I. Khaleel and A. M. Ahmed, "Green Cloud Framework for Reducing Carbon Dioxide Emissions in Cloud Infrastructure," in 2019 International Conference on Advanced Science and Engineering (ICOASE), Apr. 2019, pp. 29–34, https://doi.org/10.1109/ICOASE.2019.8723701

[6] H.-C. Chao, B. Hu, and C.-Y. Chen, "Fog computing and internet of everything for emerging enterprise information systems," Enterp. Inf. Syst., vol. 12, no. 4, pp. 371–372, Apr. 2018, https://doi.org/10.1080/17517575.2017.1405076

[7] A. Mijuskovic, R. Bemthuis, A. Aldea, and P. Havinga, "An Enterprise Architecture based on Cloud, Fog and Edge Computing for an Airfield Lighting Management System," in Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW, Oct. 2020, vol. 2020-October, pp. 63–73, https://doi.org/10.1109/EDOCW49879 .2020.00021

[8] D. Minoli, K. Sohraby, and B. Occhiogrosso, "IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems," IEEE Internet Things J., vol. 4, no. 1, pp. 269–283, Feb. 2017, https://doi.org/10.1109/JIOT.2017.2647881

[9] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog Computing: A Comprehensive Architectural Survey," IEEE Access, vol. 8, pp. 69105–69133, 2020, https://doi.org/10.1109/ACCESS.2020.2983253

[10] S. R. Leena, V. Divya and J. F. Lilian, "Intelligent Scheduling in Fog Environment based on Improved Hybrid Heuristics," 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA), Antalya, Turkey, 2020, pp. 1-7, Doi: https://doi.org/10.1109/AICCSA50499.2020.9316493

[11] Tian, F.; Zhang, D.; Yuan, Y.; Fu, G.; Li, X.; Chen, G. Fog Computing Task Scheduling of Smart Community Based on Hybrid Ant Lion Optimizer. Symmetry 2023, 15, 2206. https://doi.org/10.3390/sym15122206

[12] Anand, J.; Karthikeyan, B. Exploration of Multi-Task Scheduling in Multi-Access Edge Computing. Eng. Proc. 2024, 62, 4. https://doi.org/10.3390/engproc2024062004

[13] Ahanger, T.A.; Dahan, F.; Tariq, U.; Ullah, I. Quantum Inspired Task Optimization for IoT Edge Fog Computing Environment. Mathematics 2023, 11, 156. https://doi.org/10.3390/math11010156

[14] K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali, and D. A. Ibrahim, ‘‘A review of fog computing and machine learning: Concepts, applications, challenges, and open issues,’’ IEEE Access, vol. 7, pp. 153123–153140, 2019

[15] Sun, Z.; Li, C.; Wei, L.; Li, Z.; Min, Z.; Zhao, G. Intelligent Sensor-Cloud in Fog Computer: A Novel Hierarchical Data Job Scheduling Strategy. Sensors 2019, 19, 5083. https://doi.org/10.3390/s19235083.