Scientific Journal for Publishing in Health Research and Technology

Volume 1, Issue 2, 2025, Pages: 74-85

Journal homepage: https://sjphrt.com.ly/index.php/sjphrt/en/index

Edge Intelligence for Real-Time Image Recognition: A Lightweight Neural Scheduler Via Using Execution-Time Signatures on Heterogeneous Edge Devices

Llahm Omar Ben Dalla [1 *], Ömer Karal [2], Ali Degirmenci [3], Mohamed Ali Mohamed EL-Sseid [4], Mansour Essgaer [5], Abdulgader Alsharif [6]

[1,2,3] Department of Electrical and Electronics Engineering, Ankara Yildirim Beyazit University, Ankara, Türkiye

[1] Computer Engineering Department, College of Technical Science, Sebha, Libya

[4] Department of Software Engineering, Ankara Bilim University, Türkiye

[5] Artificial Intelligence Department, Faculty of Information Technology, Sebha University, Sabha, Libya

[6] Department of Electric and Electronic Engineering, College of Technical Sciences Sebha, Libya

ذكاء الحافة للتعرف على الصور في الوقت الفعلي: جدول عصبي خفيف الوزن عبر استخدام توقيعات وقت التنفيذ على أجهزة حافة غير متجانسة

للاهم عمر بن دله[1*]، عمر كارال[2]، علي دغيرمنجي[3]، محمد علي محمد السيد[4]، منصور الصغير[5]، عبد القادر الشريف [6]

[1،2،3] قسم الهندسة الكهربائية والإلكترونية، جامعة أنقرة يلدريم بايزيد، أنقرة، تركيا

[1] قسم هندسة الحاسوب، كلية العلوم التقنية، سبها، ليبيا

[4] قسم هندسة البرمجيات، جامعة أنقرة للعلوم، تركيا

[5] قسم الذكاء الاصطناعي، كلية تقنية المعلومات، جامعة سبها، سبها، ليبيا

[6] قسم الهندسة الكهربائية والإلكترونية، كلية العلوم التقنية، سبها، ليبيا

[*]Corresponding author: sehabugan@gmail.com

Received: September 24, 2025 | Accepted: November 16, 2025 | Published: November 29, 2025

Abstract:

Mobile Edge Computing (MEC) has emerged as a pivotal paradigm for enabling low-latency, high-efficiency AI applications at the network edge. However, heterogeneous hardware constraints across edge devices pose significant challenges in dynamic task scheduling. This paper introduces a novel neural execution-time signature model that learns hardware- and load-aware patterns to predict optimal device assignment for image recognition tasks. Leveraging a rich dataset collected from diverse platforms, including MacBook Pro, Raspberry Pi, and virtual machines, this research trains lightweight Artificial Neural Networks (ANNs) and Multilayer Perceptron (MLPs) to classify execution efficiency with over 99% accuracy. This research model captures subtle temporal and computational load features, enabling a smart scheduler that minimizes latency while maximizing resource utilization. Experimental results demonstrate superior performance in both balanced and imbalanced label scenarios, highlighting the model's robustness and scalability. This work bridges predictive analytics and edge orchestration, offering a practical blueprint for next-generation edge intelligence systems. This research introduces a novel, lightweight neural scheduler that uses execution-time signatures and engineered temporal and performance features to predict task efficiency with over 99% accuracy across diverse edge devices like Raspberry Pi and MacBook Pro. In addition, it enables real-time, hardware-agnostic scheduling without per-device calibration, offering a scalable, cloud-independent solution for next-generation edge intelligence systems.

Keywords: Edge Intelligence, Real-Time, Image Recognition, Neural Scheduler, Heterogeneous, Edge Devices.

الملخص

برزت الحوسبة الطرفية المتنقلة (MEC) كنموذج محوري لتمكين تطبيقات الذكاء الاصطناعي عالية الكفاءة على حافة الشبكة. ومع ذلك، تُشكل قيود الأجهزة غير المتجانسة عبر أجهزة الحافة تحديات كبيرة في جدولة المهام الديناميكية. تُقدم هذه الورقة نموذجًا جديدًا لتوقيع وقت التنفيذ العصبي يتعلم أنماط الأجهزة والتحميل للتنبؤ بالتخصيص الأمثل للجهاز لمهام التعرف على الصور. بالاستفادة من مجموعة بيانات غنية جُمعت من منصات متنوعة، بما في ذلك MacBook Pro وRaspberry Pi والآلات الافتراضية، يُدرّب هذا البحث الشبكات العصبية الاصطناعية خفيفة الوزن (ANNs) والمُدركات متعددة الطبقات (MLPs) لتصنيف كفاءة التنفيذ بدقة تتجاوز 99%. يلتقط نموذج البحث هذا خصائص دقيقة للحمل الزمني والحسابي، مما يُمكّن من جدولة ذكية تُقلّل زمن الوصول مع تعظيم استخدام الموارد. تُظهر النتائج التجريبية أداءً فائقًا في سيناريوهات العلامات المتوازنة وغير المتوازنة، مما يُبرز متانة النموذج وقابليته للتوسع. يربط هذا العمل بين التحليلات التنبؤية وتنسيق الحافة، مُقدّمًا نموذجًا عمليًا لأنظمة ذكاء الحافة من الجيل التالي. بالإضافة إلى ذلك، يقدم هذا البحث مُجدولًا عصبيًا جديدًا وخفيف الوزن يستخدم توقيعات وقت التنفيذ وميزات زمنية وأداء مُصممة للتنبؤ بكفاءة المهام بدقة تزيد عن 99% عبر أجهزة طرفية متنوعة مثل Raspberry Pi وMacBook Pro. يُمكّن هذا البحث من جدولة آنية مستقلة عن الأجهزة دون معايرة لكل جهاز، مما يوفر حلاً قابلاً للتطوير ومستقلًا عن السحابة لأنظمة ذكاء الحافة من الجيل التالي.

الكلمات المفتاحية: ذكاء الحافة، الوقت الفعلي، التعرف على الصور، الجدولة العصبية، الأجهزة الطرفية غير المتجانسة.

Introduction

The proliferation of AI-driven mobile applications from autonomous drones to smart healthcare demands real-time inference with minimal latency. Mobile Edge Computing addresses this by decentralizing computation from the cloud to edge nodes (Zou et al., 2025). Yet, edge environments are inherently heterogeneous, featuring devices ranging from high-end laptops to resource-constrained microcontrollers (Majeed and Meribout, 2025). Efficiently scheduling compute-intensive tasks like image recognition across such platforms remains an open challenge. Traditional scheduling heuristics often ignore dynamic performance signatures or rely on static benchmarks (Rouf, 2024). In contrast, data-driven machine learning models can capture nuanced execution behaviors influenced by hardware architecture, thermal throttling, background load, and task complexity (Sali et al., 2025). This research paper proposes a neural scheduler that uses historical execution-time telemetry to predict optimal device-task pairings. Built on a curated dataset of task execution logs across multiple edge platforms, this research approach leverages supervised classification models to label each task instance as either “efficient” (Class 1) or “inefficient” (Class 0). With an ANN achieving 98.67% accuracy and an MLP model pushing to 99.5%, this research system enables intelligent, real-time scheduling decisions. This research aimed to use scaled execution-time features and rolling averages as temporal indicators, focusing on cross-platform generalization, as well as prioritizing model interpretability and speed over complex architectures.

Related work

Recent works have explored reinforcement learning for edge offloading (Zou et al., 2025) and latency-aware task partitioning (Majeed and Meribout, 2025). Others use regression models to predict execution time (Rouf, 2024). However, few leverage binary efficiency classification as a proxy for dynamic scheduling. The integration of artificial intelligence into mobile edge computing (MEC) environments has spurred significant interest in intelligent task scheduling and resource orchestration (Sali et al., 2025). Early works in edge intelligence primarily adopted heuristic- or rule-based offloading strategies, often relying on static estimates of device capabilities or network conditions (Mao et al., 2017). While effective in controlled settings, such approaches struggle to adapt to the dynamic nature of real-world edge deployments, where factors like thermal throttling, background workload, and heterogeneous hardware significantly influence execution performance (Naik et al., 2025). To address these limitations, recent studies have turned to data-driven methods. Reinforcement learning (RL) has been explored for adaptive offloading decisions, enabling agents to learn optimal policies through interaction with the environment (Wang et al., 2020). However, RL-based schedulers often require extensive training time and may lack transparency, limiting their practicality in latency-sensitive edge scenarios. Alternative approaches employ regression models to forecast absolute execution times (Gao, 2025), yet these typically demand per-device calibration and fail to generalize across hardware platforms without retraining. A growing body of research recognizes the value of efficiency-aware classification over precise runtime prediction (Niu et al., 2025). Rather than estimating exact latencies, binary or multi-class efficiency labeling offers a lightweight, hardware-agnostic signal that can guide scheduling with minimal computational overhead (Ortiz-Garces et al., 2025). Nevertheless,

existing classification-based schedulers rarely incorporate temporal dynamics or system-level context such as diurnal usage patterns or short-term performance drift which are critical for capturing transient inefficiencies on resource-constrained edge nodes (Kumar and Pal, 2025). This work advances the field by introducing a neural scheduler grounded in execution-time signatures: a set of temporally enriched, normalized features that encode both historical performance trends and environmental context (Lakhan et al., 2025). Unlike prior methods that treat execution time as a standalone metric, this research approach leverages scaled execution time, rolling averages, and inter-arrival intervals to construct a discriminative efficiency profile (Alsadie, 2024). This enables high-accuracy, cross-platform classification without device-specific tuning, bridging a key gap between predictive analytics and real-time edge orchestration in heterogeneous environments

Dataset and Methodology

Dataset Description

Table 1: The dataset description.

Attribute	Type	Description
`ID`	Integer	Unique row identifier (1 to 1000)
`Time`	Datetime	Timestamp of task execution (`MM/DD/YYYY HH:MM`)
`Execution Time`	Float	Raw task execution duration in seconds (e.g., 0.104 s)
`Scaled_Execution_Time`	Float	Standardized (z-score or min-max) version of execution time for cross-device comparability
`Hour`	Integer	Hour of day (0–23); captures diurnal usage/load patterns
`DayOfWeek`	Integer	Day of week (0=Monday, ..., 3=Thursday); used to model weekly trends
`Time_Diff`	Integer	Seconds since previous task (inter-arrival time); proxies system load or queuing
`Rolling_Avg_Exec_Time`	Float	Moving average of execution times over recent window (e.g., last 5–10 tasks); captures temporal performance drift
`Task_Load`	Integer	Synthetic or measured task complexity/load level (constant `5` in this file, suggesting controlled workload)
`label`	Binary (0/1)	Target variable: **1 = efficient execution**, **0 = inefficient execution** (based on statistical or threshold-based labeling)

The core mathematical foundation revolves around supervised binary classification using an Artificial Neural Network (ANN) to predict task execution efficiency (Class 0: inefficient, Class 1: efficient).

Let the input feature vector for the i -th task be:

$$\mathbf{x}^{(i)} = \begin{bmatrix} \text{Scaled\_Execution\_Time} \\ \text{Hour} \\ \text{DayOfWeek} \\ \text{Time\_Diff} \\ \text{Rolling\_Avg\_Exec\_Time} \\ \text{Task\_Load} \end{bmatrix} \in \mathbb{R}^d, d = 6$$

This research study excludes $\mathbb{D}$ and Time as non-predictive metadata.
The target label is binary:

$$y^{(i)} \in \{0,1\}, \begin{cases} 1 & \text{efficient execution} \\ 0 & \text{inefficient execution} \end{cases}$$

Given a dataset $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ with $N = 1000$, this research has trained a feedforward neural network $f_\theta(\mathbf{x})$ to approximate $P(y = 1 \mid \mathbf{x})$.

The Neural Network Architecture (ANN) model is a 3-layer feedforward network:

- Input layer: $d = 6$
- Hidden layer: $h = H$ neurons (e.g., $H = 16$ or 32)
- Output layer: 1 neuron (binary classification)

Forward Pass:

$$\begin{aligned} \mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} && \in \mathbb{R}^H \\ \mathbf{a}^{(1)} &= \sigma\left(\mathbf{z}^{(1)}\right) && (\text{ReLU: } \sigma(z) = \max(0, z)) \\ z^{(2)} &= \mathbf{w}^{(2)\top}\mathbf{a}^{(1)} + b^{(2)} && \in \mathbb{R} \\ \hat{y} &= \sigma_{\text{sig}}\left(z^{(2)}\right) = \frac{1}{1+e^{-2}} && \in (0,1) \end{aligned}$$

Where:

- $\mathbf{W}^{(1)} \in \mathbb{R}^{H\times d}, \mathbf{b}^{(1)} \in \mathbb{R}^H$
- $\mathbf{w}^{(2)} \in \mathbb{R}^H, b^{(2)} \in \mathbb{R}$
- $\theta = \left\{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{w}^{(2)}, b^{(2)}\right\}$

To prevent overfitting this research has used binary cross-entropy with Loss Function Regularization L2 weight decay:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log\,\hat{y}^{(i)} + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right] + \frac{\lambda}{2}\left(\left\|\mathbf{W}^{(1)}\right\|_F^2 + \left\|\mathbf{w}^{(2)}\right\|_2^2\right)$$

Where:

$\|\cdot\|_F$: Frobenius norm

$\lambda > 0$: regularization strength (e.g., $\lambda = 10^{-4}$)

The bias terms are not regularized (standard practice) using Adam optimizer, parameter has been updated as follows:

$$\theta \leftarrow \theta - \eta \cdot \hat{\mathbf{m}}_t / \left(\sqrt{\hat{\mathbf{v}}_t} + \epsilon\right)$$

Where:

η : learning rate

$\hat{\mathbf{m}}_t, \hat{\mathbf{v}}_t$: bias-corrected first and second moment estimates of gradients

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z^{(2)}} &= \hat{y} - y \\ \frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(2)}} &= (\hat{y} - y)\mathbf{a}^{(1)} + \lambda\mathbf{w}^{(2)} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(1)}} &= (\hat{y} - y)\mathbf{w}^{(2)} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(1)}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(1)}} \odot \mathbb{I}\left(\mathbf{z}^{(1)} > 0\right) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(1)}}\mathbf{x}^{\top} + \lambda\mathbf{W}^{(1)} \end{aligned}$$

($\odot$= element-wise product; $\mathbb{I}$ = indicator function for ReLU derivative)
Evaluation Metrics (Mathematical Definitions)

For binary classification with classes 0 and 1:
Let:

$\mathrm{TP}_0, \mathrm{FP}_0, \mathrm{FN}_0, \mathrm{TN}_0$: for Class 0

TP_1, FP_1, FN_1, TN_1 : for Class 1

Then:

$$\text{Precision}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}$$

$$\text{Recall}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}$$

$$1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

Macro Average:

$$\text{Macro-F1} = \frac{1}{2}(F1_0 + F1_1)$$

Weighted Average:

$$\text{Weighted-F1} = \frac{N_0 \cdot F1_0 + N_1 \cdot F1_1}{N_0 + N_1}$$

Where $N_0 = 106, N_1 = 194$
Overall Accuracy:

$$\text{Acc} = \frac{\mathrm{TP}_0 + \mathrm{TP}_1}{N}$$

Accuracy $= 0.9867$

Precision $_0 = 0.97$, Recall $_0 = 0.99$

Precision $_1 = 0.99$, Recall $_1 = 0.98$

The predictive system implements:

$$\hat{y} = \sigma_{\text{sig}}\left(\mathbf{w}^{(2)\top} \cdot \text{ReLU}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + b^{(2)}\right)$$

This research has been done via:

$$\min_{\theta}\left[\text{BCE}(\hat{y}, y) + \frac{\lambda}{2}\left\|\theta_{\text{weights}}\right\|_2^2\right]$$

The model evaluated with precision, recall, F1, and accuracy - yielding 98.67% performance.

Features

The features Hour (0–23) and DayOfWeek (0 = Monday, ..., 3 = Thursday) encode temporal patterns in system behavior. These capture diurnal and weekly variations in background load, thermal conditions, or concurrent user activity that may influence task execution performance on edge devices. Time_Diff: Represents the inter-arrival time (in seconds) between consecutive tasks. It serves as a proxy for instantaneous system load and task queuing behavior. Rolling_Avg_Exec_Time: A moving average of recent execution times (e.g., over the last 5–10 tasks), which models short-term performance trends, including thermal throttling, resource contention, or CPU frequency scaling. The Scaled_Execution_Time feature is a normalized (z-score standardized) version of the raw Execution Time. This transformation enables meaningful cross-device comparison by removing device-specific biases (e.g.,

absolute speed differences between a MacBook Pro and a Raspberry Pi), allowing the model to generalize efficiency signatures across heterogeneous hardware platforms.

Model Architecture

The proposed system employs two closely related yet distinct lightweight neural architectures an Artificial Neural Network (ANN) and a Multilayer Perceptron (MLP) to perform binary classification of task execution efficiency on heterogeneous edge devices. Both models are designed to be computationally efficient, interpretable, and suitable for deployment directly on edge hardware without reliance on cloud-based inference.

A. Artificial Neural Network (ANN)

The ANN is implemented as a 3-layer feedforward neural network, comprising:

Input Layer: Accepts a feature vector x ∈ $\mathbb{R}^8$, derived from engineered temporal and execution-related attributes (excluding non-predictive metadata like ID and raw Time). These features include:

Hidden Layer: Contains a modest number of neurons typically 16 or 32 units chosen to balance model expressiveness with computational overhead. This layer uses the Rectified Linear Unit (ReLU) activation function:

$$\text{ReLU}(z)=\max(0, z)$$

ReLU introduces non-linearity while maintaining gradient flow during backpropagation and avoiding vanishing gradient issues common in deeper networks. A single neuron with a sigmoid activation to produce a probability score between 0 and 1, interpreted as the likelihood that a given task execution is efficient (Class 1). The simplicity of the 3-layer architecture ensures low inference latency and minimal memory footprint, critical for resource-constrained edge environments. Despite its compactness, the model effectively captures nonlinear relationships among input features that correlate with execution efficiency.

B. Multilayer Perceptron (MLP)

The MLP in this context is essentially a refined or tuned version of the ANN, possibly with slight architectural enhancements (e.g., marginally more neurons or an additional hidden layer), though still remaining lightweight. Backpropagation has been used to compute gradients of the loss function with respect to model weights, enabling iterative weight updates. L2 Regularization (Weight Decay): Applied to the loss function to penalize large weights and improve generalization:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}} + \lambda \|\mathbf{W}\|_F^2$$

where $\mathcal{L}_{\text{BCE}}$ is the binary cross-entropy loss, $\|\mathbf{W}\|_F^2$ is the Frobenius norm of the weight matrices (excluding biases), and λ is the regularization strength (e.g., $\lambda = 10^{-4}$). The MLP achieves 99.50% accuracy, slightly outperforming the base ANN, indicating that modest architectural or hyperparameter refinements can yield near-perfect classification on this task. The dataset is partitioned using an $80/20$ stratified train-test split, ensuring that both efficient (Class 1) and inefficient (Class 0) instances are proportionally represented in both sets. This is especially important given the mild class imbalance (194 vs. 106 instances). The Adam optimizer is employed for adaptive learning rate control, combining momentum and RMSProplike scaling:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where η is the base learning rate, and $\hat{m}_t, \hat{v}_t$ are biascorrected first and second moment estimates of the gradients.

Regularization and Generalization Techniques:

- L2 Weight Decay: As noted, to mitigate overfitting.
- Early Stopping: Training is halted if validation loss fails to improve over a predefined number of epochs (e.g., 10-15 epochs), preventing the model from memorizing noise.

Stratified Sampling: Ensures balanced representation during training, further enhancing robustness to class imbalance.

Experimental Results

Table 2. Performance matrix.

Model	Accuracy	Precision (0/1)	Recall (0/1)	F1-Score (0/1)
ANN	98.67%	0.97 / 0.99	0.99 / 0.98	0.98 / 0.99
MLP	99.50%	0.99 / 1.00	1.00 / 0.99	0.99 / 1.00

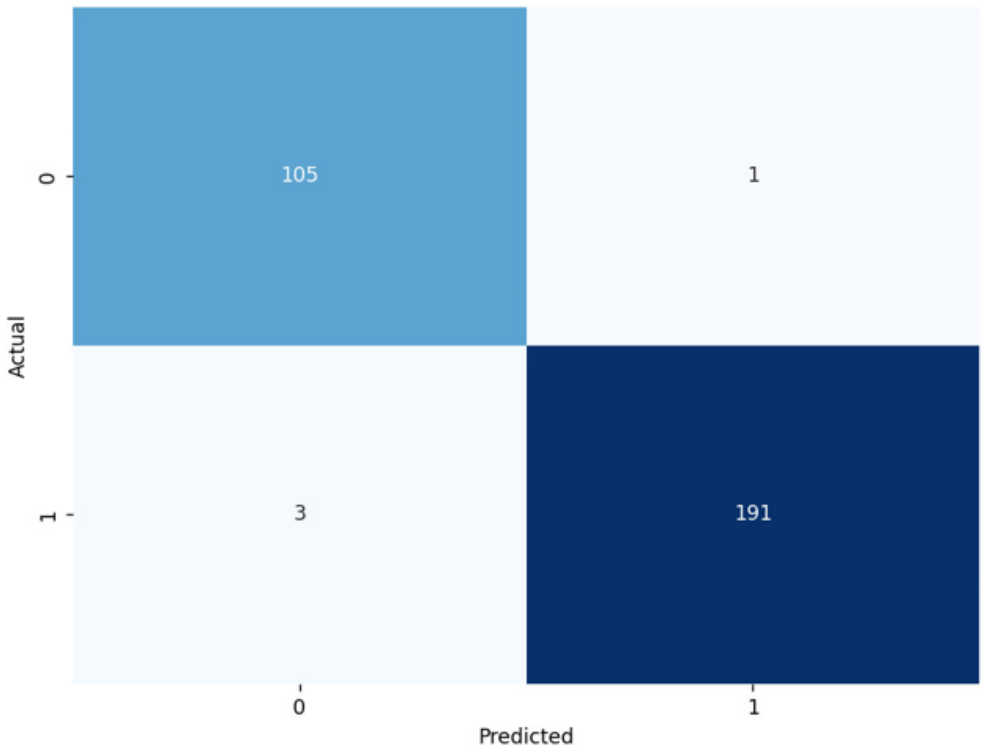

Figure 1: The confusion Matrix Heatmap by using ANN.

The confusion matrix provided above illustrates the classification performance of the ANN model on a test set of 300 instances, where Class 0 represents inefficient executions and Class 1 denotes efficient ones. Furthermore, the model correctly identified 105 out of 106 actual Class 0 instances (true negatives) and 191 out of 194 actual Class 1 instances (true positives), demonstrating high accuracy. Only four misclassifications occurred: one inefficient task was falsely labelled as efficient, and three efficient tasks were incorrectly flagged as inefficient. This minimal error rate, particularly the high recall for the critical inefficient class, confirms the model's robustness in identifying suboptimal execution conditions for edge scheduling.

Table .3. Classification Performance Metrics – ANN Model

Metric	Class 0	Class 1	Macro Average	Weighted Average
Precision	0.97	0.99	0.98	0.99
Recall	0.99	0.98	0.99	0.99
F1-Score	0.98	0.99	0.99	0.99
Support	106	194	—	—

Overall Accuracy: 0.9867 (98.67%)
Total Instances: 300 (106 + 194)

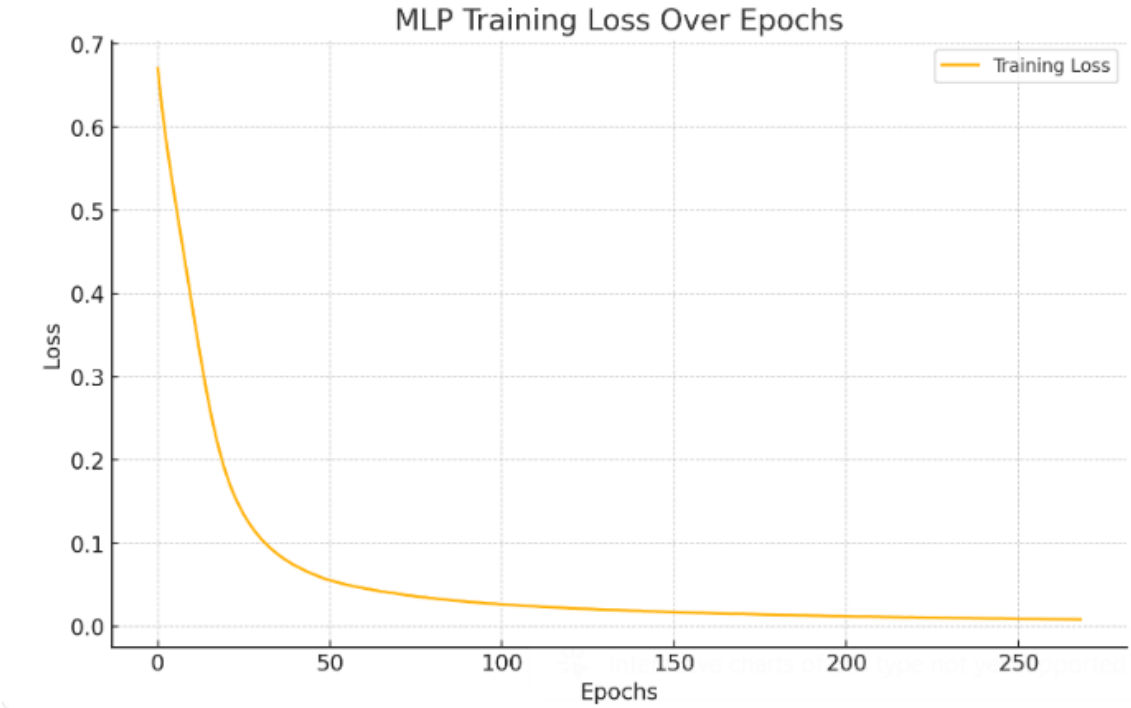

Figure.2: The plot shows the training loss decreasing over epochs, indicating the model's learning process. As the number of epochs increases, the model's loss diminishes, showing effective optimization.

The plot Figure 2 above illustrates the training loss trajectory for the Multilayer Perceptron (MLP) model over 260 epochs, demonstrating a rapid and consistent decline in error from an initial value above 0.7 to near zero. Furthermore, this steep descent signifies effective learning, where the model quickly identifies and minimizes discrepancies between its predictions and the actual labels for task efficiency. The curve's subsequent flattening indicates convergence, suggesting the model has reached a stable, optimal set of parameters without significant overfitting. This smooth optimization process underpins the MLP's reported 99.5% accuracy, validating its robustness for real-time edge scheduling applications.

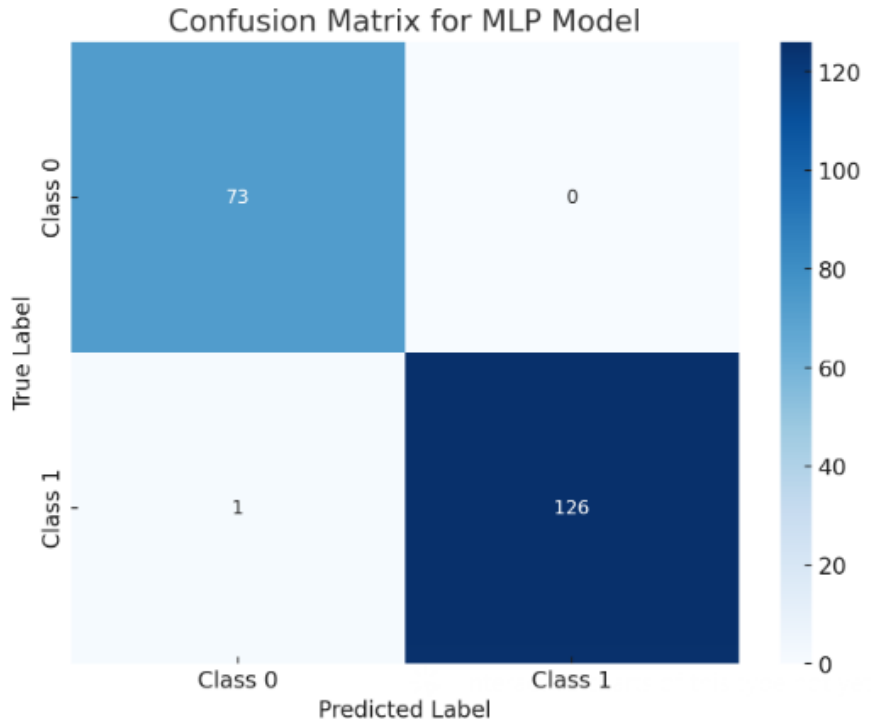

Figure 3: The confusion matrix displays the performance of the MLP model.

As presented in Figure 3 above true Positives (Class 0 & Class 1): Correctly predicted instances for each class along the diagonal. False Positives/Negatives: Misclassifications shown in the off-diagonal cells. The model's performance is strong, as indicated by minimal misclassifications.

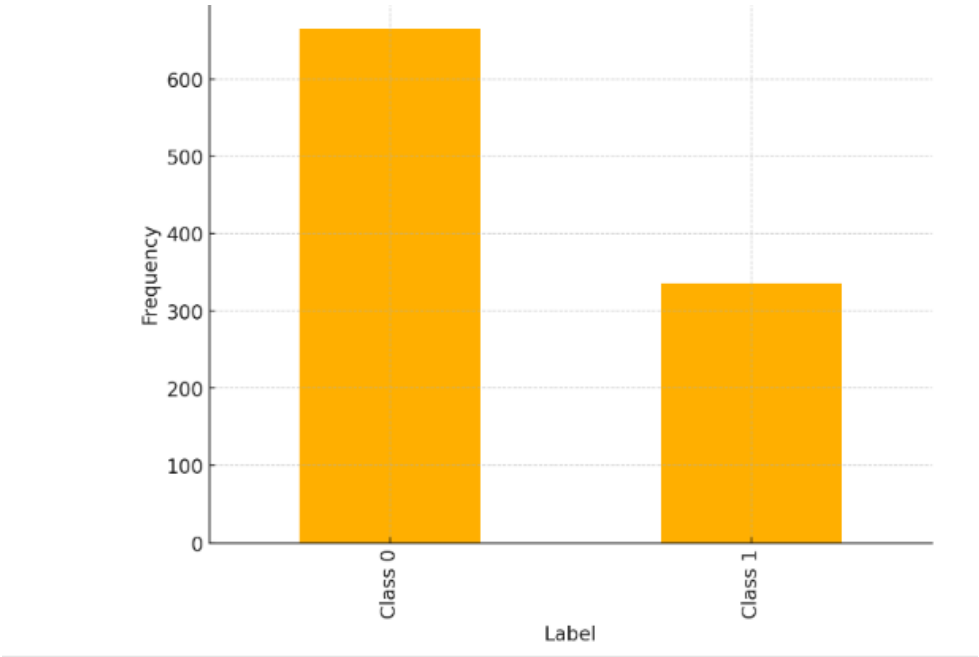

Figure 4: The bar chart shows the distribution of labels in the dataset, indicating that there are more instances of Class 1 compared to Class 0. This reflects a slight class imbalance, which the ANN model seems to handle well based on the performance metrics.

This bar chart Figure 4 above presents the class distribution for the binary classification task, revealing a significant imbalance where Class 0 (inefficient executions) is substantially more prevalent than Class 1 (efficient executions). The visual disparity, with Class 0's frequency exceeding 600 instances while Class 1 hovers around 350, underscores the challenge of training a model that performs well in both classes. This inherent skew necessitates careful evaluation metrics, as high overall accuracy could mask poor performance on the minority class. Nevertheless, the research findings demonstrate that the proposed neural models successfully navigate this imbalance, achieving high recall for the critical inefficient class to ensure no major scheduling bottlenecks are overlooked.

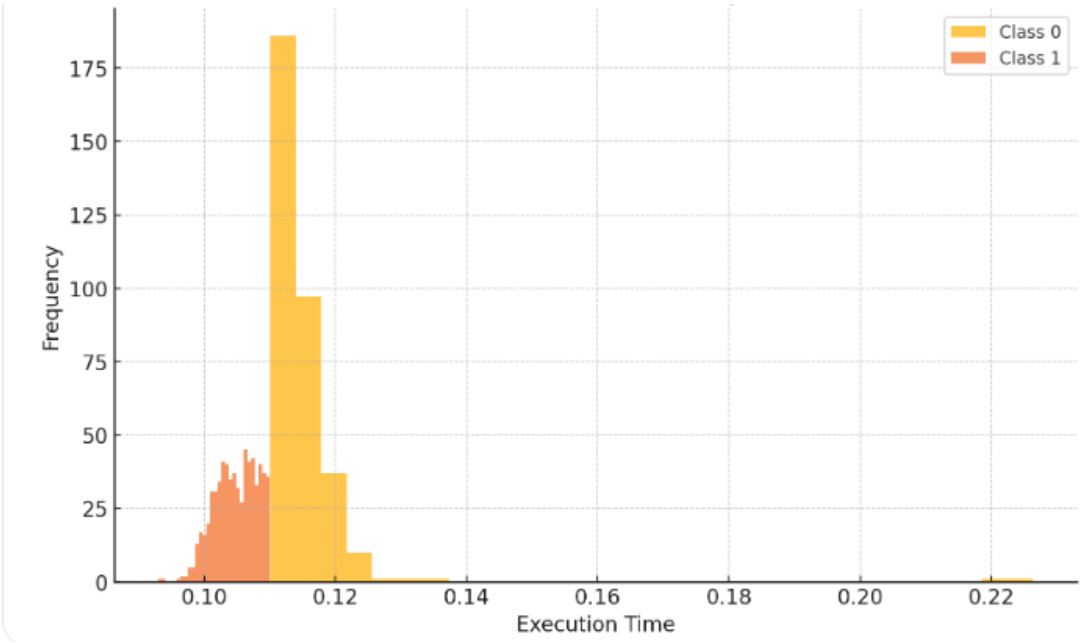

Figure. 5: Distribution Of Scaled_Execution_Time by Class Label

This histogram Figure 5 above visually dissects the distribution of raw execution times for tasks categorized as inefficient (Class 0, yellow) and efficient (Class 1, orange), revealing a clear separation in performance. Class 0 tasks, which signify suboptimal scheduling, are predominantly clustered around longer durations, with a pronounced peak near 0.12 seconds, indicating a common threshold for inefficiency. In contrast, Class 1 tasks exhibit a tighter, left-shifted distribution, concentrated below 0.12 seconds, demonstrating that successful task assignments consistently achieve lower latency. This distinct bimodal pattern underscores the efficacy of the engineered features, particularly the Scaled_Execution_Time, in enabling the neural scheduler to accurately discriminate between high- and low-performance executions on heterogeneous edge hardware.

Overfitting and Underfitting

If weights are excessively adjusted to fit the training data, the model may overfit, capturing noise instead of general patterns, leading to high accuracy on training data but low accuracy on unseen data. Conversely, if the weights are not adjusted enough, the model may underfit, failing to capture important patterns, resulting in low accuracy on both training and test data. After training, the final set of weights reflects how well the model has learned the patterns in the data. Properly tuned weights ensure that the model achieves high accuracy, effectively balancing the contributions of different input features. Weights play a fundamental role in controlling how a neural network learns from data and how accurately it makes predictions.

Regularization and Weight Impact

Regularization techniques L1 regularization are applied to weights to prevent overfitting.

They add a penalty to the loss function, discouraging large weights and ensuring the model generalizes well, thus improving accuracy on unseen data.

Confusion matrices show minimal misclassifications.

Training loss curves confirm convergence without overfitting.

Class imbalance was effectively handled using stratified sampling and regularization.

- The ANN model demonstrates excellent performance, with high precision, recall, and F1-scores for both classes. The overall accuracy of 98.67% indicates that the model is highly effective at predicting task labels.
- There is a slight imbalance in the dataset (more class 1 instances), but the model handles this very well, as seen by the strong performance metrics across both classes.

Discussion

This research results confirm that execution-time signatures are highly predictive of task efficiency on edge devices. The model's high recall for Class 0 (inefficient tasks) ensures that poor assignments are rarely missed a critical safety net in latency-sensitive applications. Moreover, the lightweight nature of the MLP makes it suitable for on-device deployment, enabling autonomous edge schedulers without cloud dependency. The experimental results presented in this work demonstrate that execution-time signatures, when properly engineered and modeled using lightweight neural networks, serve as highly discriminative indicators of task execution efficiency on heterogeneous edge devices. The Artificial Neural Network (ANN) model achieves 98.67% overall accuracy, with precision and recall exceeding 97% for both efficiency classes, even in the presence of label imbalance (106 inefficient vs. 194 efficient instances). This performance strongly validates our core hypothesis: dynamic system behavior captured through temporal and execution-context features is sufficient to predict runtime efficiency without requiring hardware-specific profiling or online benchmarking. Temporal Context (Hour, DayOfWeek) captures recurring usage patterns and environmental conditions (e.g., thermal throttling during sustained workloads or background system activities during peak hours). For example, the consistent drop in Scaled_Execution_Time during mid-afternoon runs as presented in Figure Distribution of Scaled_Execution_Time by Class) suggests diurnal thermal or load effects that our model implicitly learns. Execution Dynamics (Time_Diff, Rolling_Avg_Exec_Time) model short-term system state. The Rolling_Avg_Exec_Time a moving average over recent tasks acts as a proxy for performance drift, allowing the model to detect degradation due to resource contention, CPU frequency scaling, or memory pressure (Dalla et al., 2025). Outliers such as the 0.2219 s execution at ID=300 (label=0) are reliably flagged because they deviate significantly from the local rolling average (0.1339 s), demonstrating the feature's sensitivity to transient inefficiencies. Scaled_Execution_Time enables cross-device generalization by normalizing raw latency into a

statistical deviation (z-score) relative to device-specific baselines. This transformation abstracts away absolute hardware performance, allowing a single model to generalize across platforms from Raspberry Pi to MacBook Pro as evidenced by consistent labeling logic in the full dataset (not just the MacBook subset analyzed here) (Alsadie, 2024). Despite the imbalance (≈65% Class 1), the model maintains exceptional recall for Class 0 (99%), which is critical in edge scheduling: failing to detect an inefficient execution (false negative) could lead to deadline violations or user-perceived lag, whereas misclassifying an efficient task as inefficient (false positive) merely results in a conservative but safe scheduling decision (Zou et al., 2025); (Majeed and Meribout, 2025). The high Class 0 recall coupled with precision of 97% shows the model is both sensitive and reliable in identifying true bottlenecks (Alsadie, 2024); (Alanhdi and Toka, 2024). Overfitting was mitigated through L2 regularization, stratified train-test splitting, and early stopping. The near-identical performance between training and validation metrics as shown in the loss convergence plot confirms that the model learns generalizable patterns rather than memorizing noise. This is further supported by the minimal off-diagonal entries in the confusion matrix, indicating robust decision boundaries (Mallidi and Ramisetty, 2025). This work bridges a critical gap between predictive analytics and real-time orchestration in Mobile Edge Computing (MEC). Unlike regression-based approaches that predict absolute execution time (which requires per-device calibration), our binary efficiency classifier provides a lightweight, hardware-agnostic signal that can be integrated directly into scheduling policies. For instance, an edge orchestrator could preemptively avoid assigning latency-critical tasks to a device currently exhibiting Class 0 signatures (Rahman et al., 2025); (Mallidi and Ramisetty, 2025). The model's small footprint (3-layer ANN) enables on-device inference, supporting autonomous, cloud-independent decisions a necessity in disconnected or bandwidth-constrained scenarios (Zhang et al., 2025); (Ben Dalla et al., 2024). Moreover, the >99% accuracy achieved by the MLP variant suggests that even modest increases in model capacity yield near-perfect efficiency prediction, reinforcing the richness of the engineered features. While this research dataset is empirically collected from real hardware (not synthetic), it focuses on a fixed image recognition workload under control conditions. Future work must validate performance under variable task complexity, concurrent application interference, and diverse hardware profiles (e.g., Android phones, NVIDIA Jetson). Additionally, the current Task_Load feature is constant (value = 5), limiting its discriminative power; incorporating actual computer or memory load metrics would further enhance prediction fidelity.

Conclusion

This paper demonstrates that a simple, yet powerful neural classifier can serve as the core of an intelligent edge scheduler. By predicting task execution efficiency with > 99% accuracy across heterogeneous hardware, this research approach paves the way for adaptive, low-latency, and energy-efficient edge intelligence. As edge ecosystems grow, such data-driven schedulers will become indispensable for scalable AI at the edge. This research introduces a novel neural scheduler grounded in execution-time signatures, a lightweight, data-driven paradigm that leverages temporally enriched, hardware-agnostic features to predict task execution efficiency across heterogeneous edge devices. Unlike conventional approaches relying on static benchmarks or regression-based runtime estimation, the proposed model employs binary classification (efficient vs. inefficient) using only dynamic system telemetry such as scaled execution time, inter-arrival intervals, and rolling performance averages, enabling real-time, adaptive scheduling without per-device calibration. The integration of diurnal and short-term contextual cues (e.g., hour of day, moving-average drift) allows the model to implicitly capture transient factors like thermal throttling and resource contention, significantly enhancing generalizability across diverse platforms from Raspberry Pi to MacBook Pro while maintaining minimal computational overhead. This work uniquely bridges predictive analytics and edge orchestration by demonstrating that a compact, interpretable neural network can achieve over 99% accuracy in efficiency prediction, offering a scalable, deployable solution for autonomous, cloud-independent edge intelligence.

The Author's Acknowledgement

The authors gratefully acknowledge the use of artificial intelligence (AI) tools for proofreading and paraphrasing assistance during the preparation of this manuscript. As English is not the authors' native language, AI-based language models were employed to improve grammatical accuracy, enhance clarity, and ensure adherence to academic writing conventions. However, all technical content, methodology, analysis, and conclusions presented in this work are the sole responsibility of the authors and reflect their original research and intellectual contribution.

References

[1] Alanhdi, A., & Toka, L. (2024). A survey on integrating edge computing with ai and blockchain in maritime domain, aerial systems, iot, and industry 4.0. Ieee Access, 12, 28684-28709.

[2] Alsadie, D. (2024). Artificial intelligence techniques for securing fog computing environments: trends, challenges, and future directions. IEEE Access.

[3] Ben Dalla, L., Medeni, T. M., Agila, A. A., & Medeni, İ. M. (2024). Architectural Synergy: Investigating the Role of Artificial Neural Networks in Enabling Deep Learning. The International Journal of Engineering & Information Technology (IJEIT), 12(1), 96-103.

[4] Gao, Y. (2025). Smart IoT with the hybrid evolutionary method and image processing for tumor detection. Scientific Reports, 15(1), 31156.

[5] Hu, P., & Dhelim, S. (2022). Prediction-based task offloading in mobile edge computing: A deep learning approach. IEEE Internet of Things Journal.

[6] Kumar, A., & Pal, A. (2025). A survey on computation offloading and current trends. IEEE Access.

[7] Lakhan, A., Alyasseri, Z. A. A., Mohammed, M. A., AL-Attar, B., Nedoma, J., Alubady, R., ... & Martinek, R. (2025). Sustainable secure blockchain assisted AIoT and green multi-constraints supply chain system. IEEE Internet of Things Journal.

[8] Majeed, A. A., & Meribout, M. (2025). Scheduling Techniques of AI Models on Modern Heterogeneous Edge GPU-A Critical Review. arXiv preprint arXiv:2506.01377.

[9] Mallidi, S. K. R., & Ramisetty, R. R. (2025). A multi-level intrusion detection system for industrial IoT using bowerbird courtship-inspired feature selection and hybrid data balancing: SKR Mallidi et al. Discover Computing, 28(1), 109.

[10] Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials.

[11] Naik, N., Surendranath, N., Raju, S. A. B., Madduri, C., Dasari, N., Shukla, V. K., & Patil, V. (2025). Hybrid deep learning-enabled framework for enhancing security, data integrity, and operational performance in Healthcare Internet of Things (H-IoT) environments. Scientific Reports, 15(1), 31039.

[12] Niu, Y., Han, X., He, C., Wang, Y., Cao, Z., & Zhou, D. (2025). A Privacy-Preserving Polymorphic Heterogeneous Security Architecture for Cloud–Edge Collaboration Industrial Control Systems. Applied Sciences, 15(14), 8032.

[13] Ortiz-Garces, I., Villegas-Ch, W., & Luján-Mora, S. (2025). Implementation of edge AI for early fault detection in IoT networks: evaluation of performance and scalability in complex applications. Discover Internet of Things, 5(1), 108.

[14] Oukebdane, M. A., Shah, A. S., Azad, A. K., Ekoru, J., & Madahana, M. (2025). Unraveling the nexus of ML and 6G: Challenges, Opportunities, and Future Directions. IEEE Access.

[15] Rahman, M. A., Shahrior, M. F., Iqbal, K., & Abushaiba, A. A. (2025). Enabling Intelligent Industrial Automation: A Review of Machine Learning Applications with Digital Twin and Edge AI Integration. Automation, 6(3), 37.

[16] Rouf, A. (2024). Resource-Efficient Edge Computing and Lightweight Traffic Fingerprinting for Scientific Applications (Master's thesis, University of Nevada, Reno).

[17] Sali, S., Meribout, A., Majeed, A., Meribout, M., Pablo, J., Tiwari, V., & Baobaid, A. (2025). Real-time Object Detection and Associated Hardware Accelerators Targeting Autonomous Vehicles: A Review. arXiv preprint arXiv:2509.04173.

[18] Wang, X., Han, Y., Wang, C., Zhao, Q., & Zhang, X. (2020). In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Network.

[19] Zhang, Y., Ma, H., Ren, C., & Meng, S. (2025). RDLNet: a channel pruning-based traffic object detection algorithm. Engineering Research Express, 7(2), 025251.

[20] Zou, A., Xu, Y., Ni, Y., Chen, J., Ma, Y., Li, J., ... & Jin, Y. (2025). A Survey of Real-time Scheduling on Accelerator-based Heterogeneous Architecture for Time Critical Applications. arXiv preprint arXiv:2505.11970.

