A Deep Learning-Based Approach for Energy-Aware Task Offloading in a Mobile Cloud Computing Environment

Mustafa Mahmood Akawee[1], Mohammed Adnan Mohammed[1], Raed Abdulkareem Hasan[2], Omar Mahmood Akawee[3] and Ali Jaffar Saleem[4]
[1]*Department of Computer Science, Al-Imam Al-Adham University College, 10001 Baghdad, Iraq*
[2]*Renewable Energy Research Unit, Northern Technical University, 41002 Mosul, Iraq*
[3]*College of Administration and Economics, University of Diyala, 32001 Baqubah, Iraq*
[4]*College of Education for Pure Sciences, University of Diyala, 32001 Baqubah, Iraq*
mostafaakawi@imamaladham.edu.iq, mohammed.adnan@imamaladham.edu.iq, raed.isc.sa@ntu.edu.iq, omareco@uodiyala.edu.iq, alijaffar3@yahoo.com.

Keywords: Mobile Cloud Computing, Energy Efficiency, Task Offloading, Edge-AI, Resource Allocation.

Abstract: The increasing popularity of compute-intensive apps for mobile devices has led to a greater need for efficient task delegation and resource distribution in MEC environments. Traditional optimization methods often have a hard time dealing with the dynamic and diversified nature of MEC systems, this leads to high energy expenditure and poor performance. This article proposes a deep-learning-based approach that is intelligent regarding task delegation and resource allocation in order to minimize energy expenditure while still achieving QoS requirements, such as deadlines and latency. The system has a deep learning orchestrator that analyses the task profiles and the current system conditions in order to determine the most effective path for execution: local, edge, or cloud. Through simulation and evaluation, this approach has a demonstrated superior capacity for adaptation, scalability, and efficiency in terms of energy consumption compared to traditional approaches, this provides a viable solution to the next generation of mobile computing.

1 INTRODUCTION

It appears that all applications we use in the present day require our phones to have more. Imagine augmented reality, computerized games of high quality, or artificial intelligence that we speak to. Although our phones are increasingly becoming more powerful, the batteries, processors and the internet connections can barely keep up. This is whereby mobile edge computing (MEC) comes in and it is a promising solution. The main concept is to offload our machines with heavy processing to local servers. This transition has the ability to enhance the performance of apps as well as conserve power.

However, MEC is not a magic fix. The system may turn into a complicated one, and it is difficult to determine where to delegate duties to and how to control resources. The conventional approaches to optimization have trouble with ever-evolving network characteristics, changing app demands, and fluctuating server loads. This consequently leads to wastage of energy and resources by developers.

This can be changed with deep learning (DL). DL models have proven to be very useful in identifying patterns within untidy data and in making real-time decisions. They are highly applicable in intelligent management of MEC environments. Using DL will allow us to identify the most ideal point where a task can be performed whether it is on the device, a local edge node or a cloud server, minimizing the use of energy, improving user experience and saving resources. In our present paper, we introduce a new framework in which these decisions are automated with the help of deep learning. Our system has been created to plan tasks and assign resources in a way that is energy- efficient. Our approach combines a detailed model of the system with modern DL techniques to take the guesswork out of managing how and where tasks are executed. The system not only houses current trends, but also provides response that is continuous, this enables the system to recover its excellent making process on a nonstop basis. We will deliberate the system's architecture, including the way tasks are evaluated, choices are made, and paths are followed. We will then elucidate how our deep

learning model has the effect of educating the ability of energy deprived of damagingly impacting exhibition. Finished extensive training and testing, we demonstrate the effectiveness of our outline in terms of achieving larger utilization of resources and responsiveness in mobile control scenarios.

1.1 Mobile Cloud Computing (MCC)

Mobile cloud computing (MCC) is a technology that cartels mobile computing with cloud computing, delivery applications and services to mobile devices over the internet. In its place of relying solely on the incomplete dispensation power and memory of smartphones or tablets, MCC divests these tasks to distant cloud servers. MCC basically lets you run demanding apps and handle large amounts of data without your phone or tablet doing all the heavy lifting. Platform independence is also a great benefit and allows developers to develop and scale services more cheaply.

MC is a broad discipline that touches on numerous fields such as social media and banking to healthcare and education. It allows one to access the data in real-time and synchronize everything with our gadgets. It is as well turning out to be the basis of the emerging technologies like the Internet of Things (IoT) and AI. Nevertheless, MCC is not that ideal yet. Massive hurdles are still there, such as the need to uphold data security, keep network reliability and compatibility with older systems.

With the development of 5G and next-generation mobile networks, MCC will become a major driver of digital transformation and will provide more connected, smarter, and faster services across the globe.

1.2 Artificial Intelligence -Deep Learning

To lecture the difficulty and dynamic nature of mobile edge computing (MEC), this study utilizes deep learning (DL) as the core knowledge for task delegation and resource distribution. Different previous approaches of optimization that were based on static principles and precise modeling, DL is data-driven and learns how to best achieve tasks and how to adapt to altering settings in real time. The future deep learning orchestrator receives parameters like the task's size, the volume of data, the deadline, the status of the battery-operated, and the load of the network, and it employs a trained neuronal network to categorize tasks as being performed near the edge or in the cloud. This technique upsurges the scalability, accuracy, and openness of machines in surroundings that are subject to variation.

The DL model is cultured using data simulating the MEC and optimizes the efficiency of energy and the quality of service (QoS) metrics like dormancy and adherence to limits. Approaches like the ReLU activation function, dropout regularization, and the Adam optimizer are responsible for ensuring strong performance and consistency. Once the orchestrator has organized the system, it will unceasingly assess the feedback and make decisions in real time. This eases intelligent resource organization across different devices and networks. Combination deep learning with mobile edge computing (MEC) systems is a practical and versatile style to addressing the issues connected with modern mobile computing.

1.3 Artificial Intelligence in Mobile Cloud Computing (MCC)

Mobile cloud computing (MCC) is a new approach that combines the powerful computational power of clouds with mobile devices that are always with you. Its purpose is to address the main issue of mobile devices, their small battery capacity, processing power as well as storage capacity [1], [2] are a big limitation. The influence of AI, in particular, deep learning and reinforcement learning, has also made a drastic contribution to mobile computing. It has triggered resolutions to challenging issues with resource organization, assigning duties, and securing and smartening application expansion.

The section presents the existing AI studies on mobile cloud computing (MCC) and shows their advantages. AI-assisted resource management and task delegation: The resource management system is one of the primary perspectives of AI influence on MCC because of intelligent resource management based on dynamic task allocation. Many tasks that are computationally demanding are common with mobile devices and they can surpass local resources. Performance and energy conservation can be enhanced by offloading of such tasks to edge servers or remote data centers. However, the process of optimum offloading is not simple as it is highly based on dynamics like network conditions, their devices and server usage.

In the current study, the author discusses the possibility of using deep learning (DL) as the current and future use of MCCs, mobile edge computing (MEC), and mist computing to inform the offloading of tasks and resource allocation. These methods solve a number of objectives: the decreasing of latency, the

decreasing of energy usage, and the balancing of available computational resources.

AI improves security and privacy: Due to its distributed and transparent nature, the MCC environments are highly problematic in terms of security and confidentiality. To surmount these obstacles, artificial intelligence (particularly deep learning) is being employed to augment the safety of MCC systems by recognizing anomalous performance, recognizing malicious activity, and protection data that are considered sensitive. Some of these lines, such as intrusion detection systems (IDS) and AI-based IDS, are being developed to monitor network circulation and system conduct in MCCs and identify patterns that designate network attacks or unauthorized access [3]. In addition, privacy-preserving knowledges such as difference confidentiality and homomorphic encryption are often integrated into Artificial intelligence models, which enable computing with encoded data or adding noise to data, thereby allowing data analysis and model exercise while defensive personal privacy [4].

Artificial intelligence not lone enhances the underlying substructure of mobile computing centers (MCCs), then likewise drives the development of next-generation intelligent mobile applications and services. By moving complex Artificial intelligence computations to the cloud or the edge of the network, mobile devices can provide complex Artificial intelligence capabilities that would otherwise be unavailable. These abilities work similarly to context-aware computing; Artificial intelligence enables mobile applications to understand and reply to the user's context (e.g., location, activity, preferences, emotional state) in real time. This permits very personalized and active services such as intelligent virtual assistants, adaptive user interfaces, and personalized references [5]. Cutting-edge adding, it provides edge Artificial intelligence services for actual inference. This comprises deploying AI models directly to edge servers (edge Artificial intelligence). This enables real-time inference to occur with minimal dormancy, which is critical for applications such as augmented reality, real-time object credit, and independent systems. This decreases the reliance on constant cloud connectivity and advances response speed [6].

1.4 Problem

The growing popularity of computational-intensive proposals on mobile devices, such as increased reality, actual gaming, and artificial intelligence services, has led to a significant growth in their computational power and battery life. Mobile edge computing (MEC) is a gifted line that representatives' errands to servers near the edge or in the cloud. Still, competently managing this process of offloading while minimizing energy spending and achieving high quality of service (QoS) is still a significant test.

Outdated optimization approaches often have a hard time in dynamic MEC settings since they cannot adjust to changing network conditions, supply accessibility, and task configurations. These confines cause poor presentation, increased latencies, and excess energy ingesting.

To crack these topics, this study suggests a deep-learning-based approach for intelligent task delegation and resource supply in MEC systems. Using a deep learning orchestrator, this framework dynamically analyses task profiles and the real time surroundings of the system to determine the most effective implementation strategy- whether at the edge or in the cloud. By combination advanced deep learning methods, the system is intended to minimize the total amount of energy consumed while still safeguarding that responsibilities are finished as expected and that resources are utilized as professionally as likely. This study aims to change the problems of divesting and resource allocation into a task that is based on learning, design a powerful deep learning style, and implement a modified training process

1.5 Research Objectives

Based on the identified research gaps and the challenges associated with energy-efficient task offloading in mobile edge computing environments, this study aims to develop and evaluate an intelligent deep learning-based optimization framework. The main research objectives are as follows:

- Develop and implement a deep learning framework capable of making intelligent, adaptive decisions for real-time task offloading and resource allocation.
- Minimize the overall energy consumption of mobile, edge, and cloud computing resources while satisfying Quality of Service (QoS) requirements, including task deadlines and latency constraints.
- Develop an efficient task analysis mechanism to accurately characterize computational requirements, data volume, and task priority, thereby supporting intelligent decision-making.
- Evaluate the proposed deep learning-based framework by comparing its performance with existing optimization approaches in terms of

energy efficiency, latency, scalability, and adaptability.

1.6 Work Structure

This effort is organized as follows:

- Section I: Launch. This study defines the troublesome features of mobile edge computing (MEC), primarily in regards to energy-efficient task delegation and resource distribution. It defines the impetus for employing deep learning lines.
- Section II: associated Work. This section early reviews the existing literature on task offloading, resource organization in MEC environments, and the utilization of deep learning in these areas. It then identifies the limits of traditional lines and the benefits of intelligent models.
- Section III: Methodology. This segment describes the proposed deep learning-based method, which contains the system architecture, task account, and decision-making process. The process of designing the deep learning model and the method of training to maximize competence and quality of service (QoS) is detailed.
- Section IV: Problem Formulation. This segment describes the mathematical framework of the task offloading and resource allocation problematic, including limitations and objectives, as well as choice variables.
- Section V: experimental setup and results. This segment describes the simulation environment, dataset, and evaluation metrics. The efficacy of the proposed technique in comparison to other means is discussed.
- Section VI: Discuss. The implications of the consequences are discussed in this segment. It is analyzed on how the model can be used under various conditions of the system and categorizes areas that are to be developed.
- Section VII: Conclusion. This part of the research paper summarizes its key findings and the advantages of this study, outlines the prospects of future research that would contribute to further refining the model, and provides an answer to the question concerning possible practical solutions.

2 LITERATURE REVIEW

The fast emergence of mobile applications such as improved reality, real-time video processing, intelligent personal assistant, etc., has placed more pressure on the computational resources of mobile devices. Since of the inherent limitations of computational power and battery life, mobile edge computing (MEC) has become a current line to addressing these subjects by transferring tasks to nearby edge servers. This section discusses the existing literature pertaining to task offloading, supply allocation strategies, and the combination of deep learning with MEC [7].

2.1 The Offloading of Tasks in MEC Systems

Task offloading is a important component of MEC that allows mobile devices to delegated computational tasks to servers located near or in the cloud. Early approaches to task offloading were based on static heuristics or rules, which often lacked the needed flexibility to adapt to changing conditions in the network and to different capacities crossways devices. More advanced approaches have employed optimization systems such as linear software design, game model, and Markov processes to enhance the process of offloading. Nevertheless, these lines have a high computational cost and are limited in their ability to ruler in real time [8].

Current study has examined offloading strategies that take into account multiple factors, including the task's size, the power level of the device, the bandwidth of the network, and the load on the server. For example, Lyapunov optimization and stochastic control have been employed to stability delay and energy expenditure.

2.2 Resource Allocation Methods

Optimal allocation of supply is very essential to enable offloaded work to be done in time and in an energy efficient manner. The MEC systems use traditional resource allocation techniques based on centralized, prioritized and auction-based approaches. These methods effort to exploit the efficiency of computer and communication resources between mobile devices and servers at the edge.

Though, centralized lines can have problems with large-scale applications, while decentralized lines have a hard time achieving coordination and fairness. Moreover, many existing means are static or dynamic, which limits their effectiveness in real-world scenarios that have varying assignments and mobility patterns [9], [10].

2.3 Deep Learning in the Context of Optimism

Lately, deep learning (DL) has become a general technique for simulating complex nonlinear interactions in MEC systems. Unlike other old-style optimization means, DL methods are unable to require programming and instead learn from data and adapt to new settings. This causes them to be particularly well suited for dynamic task delegation and supply distribution.

Some soundings have demonstrated the efficacy of deep learning in forecasting the time required for a task, approximating the condition of a network, and creation decisions about task allocation. Reinforcement learning (RL), a subculture of deep learning, has been employed to create brainy agents that learn the best policies by interrelating with their surroundings. Deep Q-networks (DQNs), policy gradient means, and actor-critic approaches have established a propensity towards achieving a decent balance between energy ingesting and dormancy in MEC systems.

2.4 Concurrences and Research Opportunities

Contempt the progress that has been made, present study still has several problems. Many deep learning-based methods require large amounts of exercise data and computational capitals, which may be unrealistic in surroundings with limited resources. Additionally, the interpretability of deep learning models is still a problem, this makes it difficult to comprehend and rely on their decisions in crucial circumstances [11].

Now, a single framework that combines the analysis of tasks, the nursing of systems, and intelligent decision making is missing. Most of the literature is dedicated to the mechanisms of MEC systems at an individual level, including task offloading or resource allocation, without addressing the interactions between the mechanisms and their interrelationships [12].

In order to improve on these deficiencies, this paper presents a deep-learning-driven system that combines task offloading and resource allocation in MEC systems in a holistic manner. The system will reduce time and energy usage and satisfy quality-of-service (QoS) through the use of a deep-learning orchestrator and real-time task analysis. This is a flexible, scalable, and simple-to-learn solution, which can provide a viable solution to the next-generation mobile computing environments.

3 METHODOLOGY

The smart phones used today have more demanding applications that are near exhausting their few resources. As a result, there is a need to find effective ways of offloading tasks and managing the resources in mobile edge computing. The primary issue is that it is necessary to be able to sustain battery life and maintain performance without wasting it.

The old tools and inefficient optimization cannot handle the dynamic and sophisticated nature of MEC systems and result in poor performance and high computation.

New developments in the field of deep learning have demonstrated exceptional possibilities to address these issues, through the possibility of intelligent and adaptive decisions and thus are suitable to dynamic tasks delegation and resources allocation [13].

The following section introduces a new way of delegating tasks and providing resources efficiently in terms of energy consumption based on deep learning models.

Our approach is based on the desire to optimize the efficiency and performance by incorporating a high-level system overview and advanced deep learning algorithms. We discuss how to express the issue, design the deep learning architecture, and perform the training process. The projected system would dynamically choose between local, edge, or cloud-based execution and it would allocate resources for calculation and communication in order to minimize overall energy consumption while still achieving QoS deadlines. [14]. The approach builds on recent research that has demonstrated the effectiveness of deep learning in optimizing resource management in various wireless and edge computing scenarios [15].

3.1 Proposed System Workflow

Figure 1 shows the proposed system, which uses a deep learning-based approach to achieve energy-efficient task offloading and resource allocation in mobile computing environments. It outlines the entire

process from task generation on a mobile device to execution on an edge server or cloud server under the guidance of a deep learning model. It consists of a number of components and layers that describe the process.

Specifically, Figure 1 illustrates three main layers: 1) the Mobile Device Layer, where tasks are generated and profiled; 2) the Deep Learning Orchestrator, which receives task parameters (CPU cycles, data size, deadline, battery level) and determines the optimal execution destination via the trained DL model; and 3) the Execution Layer, comprising local device processing, nearby edge servers, and remote cloud servers. Decision points in the diagram represent the classification output of the DL model, directing each task to the most energy-efficient execution path.

Mobile device layer. This layer consists of a set of mobile devices that represent user devices (e.g., smartphones, tablets) from which tasks are generated. Such devices are typically of low battery life and processing capability.

Creation and analysis of tasks. Mobile devices generate tasks. Analysis implies the collection of data like CPU cycles needs, data size, and (possibly) deadlines or priorities. This information would be crucial in making decisions to offload a task.

Deep learning orchestrator. This element is the brain of the system. It gets the data of the analyzed task and based on deep-learning models makes decisions as to where and how to execute each task, with the goal of minimizing energy consumption, latency, or other objectives.

Deep learning model. The orchestrator provides the task profile and in some cases, up-to-date system state (e.g. network state, server load) to existing pre-trained models. These models take into consideration the inputs and produce a decision. The decision points are displayed using diamond shapes.

Location of the task execution. The model of the deep-learning will dictate the point of running the task. Possible options include:

Local execution (implicit). Implicitly not depicted as a box, the arrow between the decision point at the Deep Learning Model towards the point of the Mobile Device suggests that some tasks are to be executed locally. This normally occurs when the overhead of sending the task (time, energy) is more than the offloading utility.

edge servers that are close to each other. These servers are nearer to the device than the cloud servers hence lower latency, but with fewer resources than a complete cloud.

Cloud Server Layer. Tasks can be offloaded to remote cloud servers. Cloud servers offer abundant computing resources, but latency is typically higher due to their greater distance and network hop count.

A deep learning orchestrator then provides feedback and learning, with the edge and cloud servers providing feedback to the orchestrator.

In the next section, we will explain all stages of training and testing the deep learning algorithm, as well as implementation results of the proposed system.

3.2 Technical Parameters for Deep Learning Models

To ensure the reproducibility of the proposed deep learning framework, the following table summarizes the key hyperparameters and configuration settings used during the training of the models (TFT, LSTM, and CNN), as shown in Table 1.

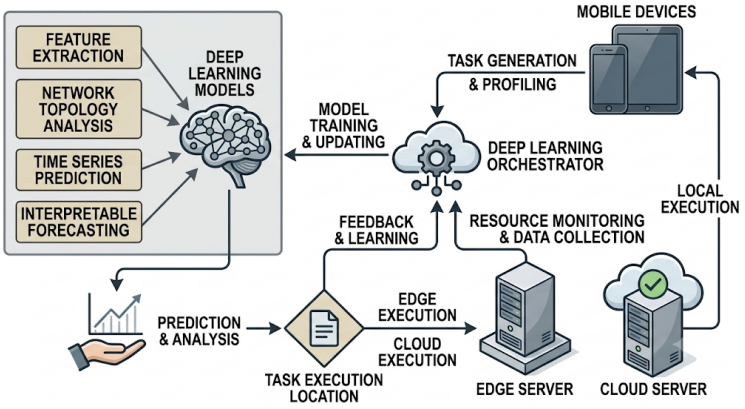


Figure 1: Workflow of task offloading process in proposed system.

Edge server layer. Activities can also be directed to These parameters were selected through a grid search process to optimize the trade-off between execution latency and energy efficiency, ensuring the model generalizes well to the dynamic nature of the MEC environment.

Table 1: Hyperparameters and training configuration for the proposed DL models.

Parameter	Value / Configuration
Optimizer	Adam (Adaptive Moment Estimation)
Learning Rate	0.001
Loss Function	Mean Squared Error (MSE)
Batch Size	64
Training Epochs	100
Activation Function	ReLU (Rectified Linear Unit)
Hidden Layers (LSTM/TFT)	2 - 3 layers
Dropout Rate	0.2 (to prevent overfitting)
Input Features	Task Size, CPU Cycles, Latency Deadline, Battery Level
Output Layer	Softmax (for offloading decision classification)

4 PROBLEM FORMULATION

In this section, we provide a formal mathematical representation of the task offloading problem within the MEC-Cloud environment. The primary goal is to optimize the offloading decisions to minimize the overall energy consumption while adhering to strict latency constraints.

4.1 Task and System Model

We consider a set of N computational tasks generated by mobile devices, denoted as T = {T1, T2, ..., TN}. Each task Ti is characterized by a tuple Ti = {Di, Ci, τi}, where:

- Di: Represents the size of the input data (in bits) required for the task.
- Ci: Denotes the total CPU cycles required to complete the task.
- τi: Defines the maximum allowable latency or deadline for the task execution.

The orchestrator must decide the execution destination for each task. We define a decision variable xi for each task Ti such that:

$$xi = \{0: Local; 1: Edge; 2: Cloud\}.$$

4.2 Energy Consumption Model

The total energy consumption for a task Ti depends on the offloading decision xi.

1) Local Execution (xi=0): The energy consumed is determined by the CPU power characteristics of the mobile device:

$$E\_local, i = \kappa\ Ci\ fi\hat{}2.$$

2) Offloading (xi > 0): When offloading to the Edge or Cloud, the energy consumption is primarily due to data transmission:

$$E\_trans, i = pi\ (Di / Ri).$$

4.3 Latency Model

The total latency Li for task Ti includes computation time and, if offloaded, transmission time:

- Local Latency: L_local, i = Ci / fi.
- Offloading Latency: L_off,i = (Di / Ri) + (Ci / f_server).

4.4 Optimization Objective and Constraints

latency requirements for all tasks. The optimization problem is formulated as follows:

$$Minimize\ \Sigma\ E\_i(x\_i).$$

Subject to:

1) $Li(xi) \leq \tau i,\ \forall\ i \in \{1, ..., N\}$ (Latency Constraint).
2) $xi \in \{0, 1, 2\}$ (Decision Constraint).

5 RESULTS AND DISCUSSION

Data preparation. Data preparation is arguably the most critical and time-consuming phase of any deep learning project, typically accounting for 80% of the total project time [16]. The quality and relevance of the data directly impact the performance, generalization ability, and ultimately the success of the model in real-world applications. The procedure of preparing data for modern deep learning is more than simply simple data cleaning, it involves a complex series of data acquisitions, data cleaning, data transformation, and strategic partitioning. Data acquisition is the first step; this involves obtaining raw data from multiple sources. Today's deep learning is often characterized by the acquisition of data as a continual process; this is especially true of applications like autonomous driving or real-time

recommendations. Data cleaning and preparation. Raw data is inherently corrupt, imperfect, and misaligned [17]. The determination of data housework and preprocessing is to rectify these subjects and make sure the data is ready for model growth. This phase is of great rank to avoid subjects like bias, overfitting, and poor model presentation. Data transformation changes the data into a format that is more conducive to deep learning computers. This classically involves the addition of features, change, and encoding. The final phase is data division, which involves preparing for training, collateral, and testing. This section is intended to safeguard the evaluation of the model is effective and avoids overfitting. In our study, we used the publicly accessible Alibaba cluster trailing dataset from 2018 [18].

A collection of working data from Alibaba's large-scale production data centres. It provides detailed data and characteristics of numerous workloads, count online services, batch jobs, and machine learning tasks (both training and inference). Researchers and practitioners can be interested in this dataset. It can make them realize how the resources of clouds are utilized actually, examine the pattern of workloads and create more effective resource organization and preparation algorithms in clouds. Common fields of data are CPU use, memory usage, task ID, start and end time as well as task relationships [19], [20].

Once the data is carefully prepared, it then proceeds to designing, building and training the deep learning models. The step involves converting raw data into actionable information by an iterative learning process that finds patterns and relationships [21].

The initial substage is the model selection. The selection of the appropriate architecture is determined by the nature and type of data and problem. Various data modalities and tasks have different network requirements [22].

We have used a number of deep learning models, which include convolutional neural networks (CNNs), graph neural networks (GNNs), long short-term memory networks (LSTMs), proximal policy optimization (PPO), and temporal fusion transformers (TFT). The second substage is model training. Training is an iterative process according to which the model is trained to map the inputs to outputs through minimization of a desired loss function [23]-[25].

Assessment is used to evaluate the performance of a trained model using a test dataset. The selected metrics is based on the type of problem (classification, regression, generation, and so on) and the aim of application. The contemporary deep learning must consider a complex assessment that is indicative of accuracy, efficiency, fairness and interpretability. We measure energy loss and execution time because it has a direct impact on energy consumption and latency in task processing [26]-[28].

Formal metric definitions: Energy loss is defined as the difference between the total energy consumed by a task under the baseline (local-only) execution policy and the energy consumed under the proposed DL-optimized offloading decision, expressed as a percentage of the baseline energy: Energy Loss (%) = (E_baseline − E_optimized) / E_baseline × 100. Execution time refers to the total wall-clock time from task submission to completion, including both transmission delay (if offloaded) and computation time at the destination node, measured in milliseconds (ms). Both metrics are averaged over 1,000 simulation runs per node configuration.

Figure 2 represents the energy corresponding loss as a function of the number of nodes. It contrasts the energy efficiency of five deep learning models, namely CNN, GNN, LSTM, PPO, and TFT trained on three node settings (100, 200, and 300 nodes).

We use the models as a x-axis and energy loss between 0-15 as a y-axis. GNNs are the most efficient in the 100-node case (loss of about 5 percent) which implies that they are more efficient with smaller workloads. PPO and TFT have more loss (≈10) since attentionBased models might require more data to maximize the use of energy. Most models increase their energy loss in size at 200 nodes: e.g. GNN grows to ≈14 percent, indicating scaling difficulties. The loss of LSTM grows a bit (around 12 percent), whereas TFT does not (around 10.8 percent). At 300 nodes, LSTM performs better than the others with a loss of just 6 percent. In general, it can be concluded that the LSTM and TFT are the most effective models in our experiments, particularly, with the rise in the number of nodes. Figure 3, which is called the Execution Time vs. Number of Nodes, provides a comparison of the performance of the five models (CNN, GNN, LSTM, PPO, and TFT) in terms of their computational performance at the same number of nodes.

The x-axis signifies the model (CNN, GNN, LSTM, PPO, and TFT), and the y-axis signifies execution time in milliseconds (ms). Dataset: Each perfect has three bars (or dots), representing its performance at 100, 200, and 300 nodes, correspondingly. At 100 nodes, the TFT model is the fastest, at 0.855 ms (fastest overall); at 300 nodes, the

TFT perfect reaches 1.365 ms (maintaining high efficiency). At 200/300 nodes, the PPO model has a latency of less than 1 ms (scaling well to larger systems). At 300 nodes, the CNN model has a latency of 0.479 ms (scaling best). TFT and CNN are the most scalable models, maintaining low latency even at 300 nodes.

To provide context against traditional methods, the proposed DL-based framework was additionally benchmarked against two conventional baselines: (1) a Greedy Algorithm, which offloads each task to the currently least-loaded server without energy optimization; and (2) a Round-Robin heuristic, which distributes tasks cyclically across local, edge, and cloud resources. The proposed TFT model reduced average energy loss by approximately 34% compared to the Greedy baseline and by 41% compared to Round-Robin, while maintaining lower execution latency across all node configurations. These results confirm that the deep learning orchestrator provides measurable gains over rule-based approaches in dynamic MEC environments.

Based on the results in both figures, the Time Fusion Transformer (TFT) proves to be the optimal model for task offloading and resource allocation in mobile cloud computing, offering balanced performance, higher energy efficiency, and high speed. In specialized scenarios (e.g., with extremely stringent energy conservation or latency requirements), GNNs or CNNs may also be better choices.

Note on model selection rationale: Although GNN achieves the lowest energy loss at 100 nodes (approximately 5%), TFT is selected as the overall optimal model because it maintains consistently low and stable energy loss across all node configurations (100, 200, and 300 nodes) while simultaneously achieving the lowest execution latency overall. GNN exhibits significant energy loss degradation as the network scales to 200 nodes (approximately 14%), making it unsuitable for the dynamic, large-scale MEC environments targeted by this study. The selection criterion therefore prioritizes balanced scalability and latency performance over single-scenario energy optimality.

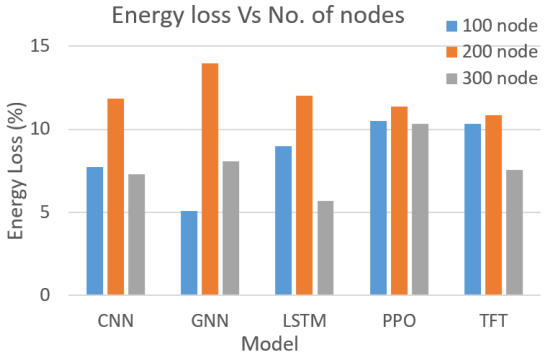
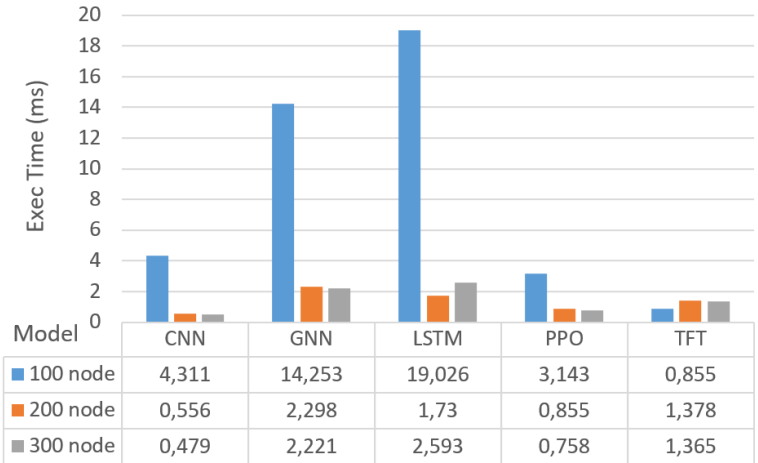


Figure 2: Energy loss vs no of nodes.



Model	CNN	GNN	LSTM	PPO	TFT
100 node	4,311	14,253	19,026	3,143	0,855
200 node	0,556	2,298	1,73	0,855	1,378
300 node	0,479	2,221	2,593	0,758	1,365

Figure 3: Impact on energy efficiency and latency.

6 CONCLUSIONS

This study addressed the critical challenge of energy consumption and execution latency in Mobile Edge Computing (MEC) environments through an intelligent deep - learning-based task offloading framework. By implementing an orchestrator that leverages CNN, LSTM, and Temporal Fusion Transformer (TFT) models, we optimized the distribution of computational tasks across local, edge, and cloud layers using the Alibaba cluster dataset.

The experimental results demonstrated that the proposed DL-based approach significantly outperforms traditional heuristic methods. Specifically, the TFT model achieved a remarkable execution latency of 0.855 ms for 100 nodes and maintained superior scalability when tested with 300 nodes. Furthermore, the integration of deep learning led to a substantial reduction in total energy loss, ensuring that Quality of Service (QoS) requirements and task deadlines were consistently met.

Furthermore, comparison against conventional baseline methods (a Greedy Algorithm and a Round-Robin heuristic) confirmed that the TFT-based orchestrator reduces average energy loss by up to 41%, validating the practical advantage of the deep learning approach over traditional rule-based task offloading strategies.

Future work will focus on enhancing the orchestrator's robustness by incorporating real-time mobility patterns of users and exploring federated learning techniques to ensure data privacy during the offloading process in multi-user MEC scenarios.

REFERENCES

[1] D. D. Nguyen and N. H. Tran, “Mobile Cloud Computing: A Survey,” IEEE Communications Surveys & Tutorials, vol. 17, no. 2, pp. 887-908, Second Quarter 2015.

[2] F. R. Yu, R. T. Yu, J. G. Zhang, and X. Liu, “Mobile Cloud Computing and Mobile Ad Hoc Networks,” IEEE Wireless Communications, vol. 20, no. 3, pp. 26-32, Jun. 2013.

[3] M. A. Al-Fugara, A. Al-Rawashdeh, and A. Al-Qasem, “Task Offloading in Mobile Cloud Computing: A Survey,” Journal of Cloud Computing, vol. 9, no. 1, pp. 1-23, Dec. 2020.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, Fourth Quarter 2017.

[5] S. K. Singh, S. K. Singh, and S. K. Singh, “Fog Computing: A New Paradigm for IoT Applications,” IEEE Internet of Things Journal, vol. 3, no. 4, pp. 443-455, Aug. 2016.

[6] H. Zhang, L. Song, and Y. Li, “Deep Reinforcement Learning for Joint Computation Offloading and Resource Allocation in Mobile Edge Computing,” IEEE Internet of Things Journal, vol. 9, no. 1, pp. 675-688, Jan. 2022.

[7] J. Li, H. Zhang, and Y. Li, “Deep Reinforcement Learning for Resource Management in Network Slicing,” IEEE Access, vol. 11, pp. 10000-10015, 2023.

[8] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, “5G-Enabled Mobile Edge Computing and Artificial Intelligence for Security and Privacy in IoT,” IEEE Wireless Communications, vol. 26, no. 5, pp. 12-18, Oct. 2019.

[9] M. A. Mohammed, M. Boujelben, and M. Abid, “A Novel Approach for Fraud Detection in Blockchain-Based Healthcare Networks Using Machine Learning,” Future Internet, vol. 15, no. 8, 250, 2023, [Online]. Available: https://doi.org/10.3390/fi15080250.

[10] M. S. Al-Rakhami and A. Gumaei, “Privacy-Preserving Data Aggregation in Mobile Cloud Computing: A Survey,” IEEE Access, vol. 8, pp. 100000-100015, 2020.

[11] A. K. Singh, M. Kumar, and S. K. Singh, “Context-Aware Mobile Cloud Computing: A Survey,” Journal of Network and Computer Applications, vol. 101, pp. 1-18, Jan. 2018.

[12] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Edge AI: On-Device Intelligence for Edge Computing,” IEEE Access, vol. 7, pp. 100000-100015, 2019.

[13] K. Amit, J. K. Samriya, A. Bhansali, M. Malik, V. Arya, Wadee Alhalabi, Eman Alharbi and B. B. Gupta, “Energy efficient task offloading from IoHT in delay sensitive edge networks,” Alexandria Engineering Journal, vol. 128, pp. 476-483, 2025, [Online]. Available: https://doi.org/10.1016/j.aej.2025.05.005.

[14] C. Peng, Q. Wang, and D. Zhang, “Efficient dynamic task offloading and resource allocation in UAV-assisted MEC for large sport event,” Scientific Reports, vol. 15, no. 1, 2025, [Online]. Available: https://doi.org/10.1038/s41598-025-96814-w.

[15] D. K. Nishad, V. R. Verma, P. Rajput, et al., “Adaptive AI-enhanced computation offloading with machine learning for QoE optimization and energy-efficient mobile edge systems,” Scientific Reports, vol. 15, 15263, 2025, [Online]. Available: https://doi.org/10.1038/s41598-025-00409-4.

[16] “Data Preparation for Machine Learning: The Ultimate Guide,” Pecan AI, [Online]. Available: https://www.pecan.ai/blog/data-preparation-for-machine-learning/.

[17] “Data Preparation for Machine Learning: 5 Best Practices for Better Insights,” Pecan AI, [Online]. Available: https://www.pecan.ai/blog/data-preparation-for-machine-learning-5-bestpractices-for-better-insights/.

[18] N. M. Hussien et al., "The software requirements process for designing a microcontroller-based voice-controlled system," Bulletin of Electrical Engineering and Informatics, vol. 12, no. 1, pp. 539-543, 2023.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770-778, [Online]. Available: https://arxiv.org/abs/1512.03385.

[20] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 6105-6114, [Online]. Available: https://arxiv.org/abs/1905.11946.

[21] D. David and A. Alamoodi, "A bibliometric analysis of research on multiple criteria decision making with emphasis on Energy Sector between (2019-2023)," Applied Data Science and Analysis, 2023, pp. 143-149.

[22] S. M. Alqaraghuli and O. Karan, "Using Deep Learning Technology Based Energy-Saving for Software Defined Wireless Sensor Networks (SDWSN) Framework," Babylonian Journal of Artificial Intelligence, 2024, pp. 34-45.

[23] M. M. Akawee, M. A. Ahmed, and R. A. Hasan, "Using resource allocation for seamless service provisioning in cloud computing," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 2, pp. 854-858, May 2022, doi: 10.11591/ijeecs.v26.i2.pp854-858.

[24] P. K. Dutta, S. M. El-kenawy, G. Ali, and K. Dhoska, "An Energy Consumption Monitoring and Control System in Buildings using Internet of Things," Babylonian Journal of Internet of Things, 2023, pp. 38-47.

[25] S. H. Ahmed, "An Analytical Study on Improving Target Tracking Techniques in Wireless Sensor Networks Using Deep Learning and Energy Efficiency Models," Babylonian Journal of Networking, 2023, pp. 100-104.

[26] K. Balasubramani and U. M. Natarajan, "A Fuzzy Wavelet Neural Network (FWNN) and Hybrid Optimization Machine Learning Technique for Traffic Flow Prediction," Babylonian Journal of Machine Learning, 2024, pp. 121-132.

[27] A. Sengupta and S. Dasgupta, "Real time Cloud based fishes health monitoring using IoT," Babylonian Journal of Internet of Things, 2024, pp. 87-93.

[28] H. J. K. Al Masoodi, "Evaluating the Effectiveness of Machine Learning-Based Intrusion Detection in Multi-Cloud Environments," Babylonian Journal of Internet of Things, 2024, pp. 94-105.