Journal of Applied Engineering and Technological Science
Vol 4(2) 2023: 1038-1056

PERFORMANCE ANALYSIS OF TASK OFFLOADING IN MOBILE EDGE CLOUD COMPUTING FOR BRAIN TUMOR CLASSIFICATION USING DEEP LEARNING

R. Yamuna[1*], Rajani Rajalingam[2], M. Usha Rani[3]
Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati, India[13]
Department of Computer Science and Engineering, Geethanjali Institute of Science and Technology, Nellore, India[2]
ryamunaspmvv@gmail.com[1], rajani@gist.edu.in[2], musha_rohan@yahoo.com[3]

Received : 22 March 2023, Revised: 07 June 2023, Accepted : 20 June 2023
**Corresponding Author*

ABSTRACT
The increasing prevalence of brain tumors necessitates accurate and efficient methods for their identification and classification. While deep learning (DL) models have shown promise in this domain, their computational demands pose challenges when deploying them on resource-constrained mobile devices. This paper investigates the potential of Mobile Edge Computing (MEC) and Task Offloading to improve the performance of DL models for brain tumor classification. A comprehensive framework was developed, considering the computational capabilities of mobile devices and edge servers, as well as communication costs associated with task offloading. Various factors influencing task offloading decisions were analyzed, including model size, available resources, and network conditions. Results demonstrate that task offloading effectively reduces the time and energy required to process DL models for brain tumor classification, while maintaining accuracy. The study emphasizes the need to balance computation and communication costs when deciding on task offloading. These findings have significant implications for the development of efficient mobile edge computing systems for medical applications. Leveraging MEC and Task Offloading enables healthcare professionals to utilize DL models for brain tumor classification on resource-constrained mobile devices, ensuring accurate and timely diagnoses. These technological advancements pave the way for more accessible and efficient medical solutions in the future.
Keywords: *Brain Tumor Classification, Deep Learning Models, Mobile Edge Computing, Task Offloading, Resource-Constrained Mobile Devices*.

1. Introduction

Brain tumors are a serious medical condition that affects a significant portion of the population. According to the American Brain Tumor Association, about 0.22% of the US population is expected to be diagnosed with a primary brain tumor in (Association., 2021). Additionally, approximately 0.21% of the US population are currently living with a primary brain tumor. Brain tumors are abnormal growths in the brain that can be classified as either benign or malignant. These tumors can develop from various types of brain cells, such as glial cells, meninges, & neurons. The symptoms of brain tumors can vary depending on their location, size, and type. For instance, about 60% of people with brain tumors experience persistent or severe headaches, while about 30% experience seizures. Memory problems and difficulty with movement or coordination are also common symptoms in (Association., 2021).

Background

The rising incidence of brain tumors has created a pressing need for accurate and efficient methods of identification and classification. Deep learning (DL) models have shown promising results in this area, particularly in medical imaging tasks like brain tumor classification. However, their implementation on resource-constrained mobile devices presents significant challenges due to the computational demands involved.

Mobile devices, such as smartphones and tablets, are widely utilized in healthcare settings due to their convenience and portability. They offer a potential solution for on-the-go medical image analysis and decision-making. However, the limited computational capabilities and battery life of mobile devices pose obstacles to deploying DL models for complex tasks such as brain tumor classification.

To address these challenges, researchers have turned to Mobile Edge Computing (MEC) and Task Offloading. MEC leverages the computational resources of nearby edge servers, which possess greater processing power, to offload tasks from mobile devices. Task offloading involves transferring computational tasks from the mobile device to the edge server for processing, thereby reducing the strain on the limited resources of the mobile device.

This study aims to explore the potential of MEC and Task Offloading in improving the performance of DL models for brain tumor classification on resource-constrained mobile devices. By harnessing the computational capabilities of edge servers, the researchers seek to minimize the time and energy required for DL model processing without compromising accuracy. Additionally, the study investigates various factors that influence the decision to offload tasks, including model size, available resources, and network conditions.
In their study,

In recent years, researchers and scientists have made significant advancements in the field of medical image analysis, particularly in the area of brain tumor classification and segmentation. They have been exploring the use of deep learning techniques, specifically convolutional neural networks (CNNs), to tackle these challenges.

One study by (Bhima K, 2017) focused on developing a powerful model using deep CNNs to accurately classify brain tumors. By analyzing MRI images, their model could effectively categorize different types of tumors with high accuracy.

Another important contribution came from (Litjens, 2017), who conducted a comprehensive survey on deep learning in medical image analysis. They discussed the potential of deep learning techniques, including CNNs, to improve accuracy and automation in interpreting medical images. The survey covered various applications, including tumor detection and segmentation.

The ImageNet Large Scale Visual Recognition Challenge (Russakovsky, 2015) played a crucial role in advancing computer vision, including medical image analysis. The challenge encouraged researchers worldwide to develop models capable of recognizing and classifying a large-scale dataset of images. The competition demonstrated the effectiveness of CNNs and other deep learning approaches in achieving state-of-the-art performance in visual recognition tasks. (Tajbakhsh, 2016) Investigated the use of CNNs for medical image analysis, specifically focusing on whether it was better to train models from scratch or fine-tune pre-existing models. They explored the performance of CNNs in tasks such as tumor detection and provided insights into the advantages and limitations of different training approaches.

(Gumaei, 2019) proposed a hybrid feature extraction method combined with a regularized extreme learning machine for brain tumor classification. Their approach aimed to improve classification accuracy by extracting relevant features from MRI images and utilizing a robust learning algorithm. Their results demonstrated the effectiveness of their method in accurately distinguishing between malignant and benign brain tumors. (Dandıl, 2015) worked on computer-aided diagnosis of brain tumors using MR images. They developed a system to assist medical professionals in differentiating between malignant and benign tumors. By employing advanced image analysis techniques, their goal was to provide reliable diagnostic support.

(Pereira, 2016) focused on brain tumor segmentation using CNNs in MRI images. They aimed to create an automated and accurate method to identify and delineate tumor regions in medical images. By training CNNs on annotated data, they achieved promising results in segmenting brain tumors, showcasing the potential of deep learning in enhancing medical image analysis.

Overall, these studies highlight the significant progress made in using deep learning techniques, particularly CNNs, for brain tumor classification, segmentation, and computer-aided diagnosis. Their findings contribute to improving diagnostic accuracy and enhancing patient care in the field of medical imaging.The findings of this research have noteworthy implications for the development of efficient and effective mobile edge computing systems tailored for medical applications. By optimizing the utilization of MEC and Task Offloading, healthcare professionals can effectively utilize DL models for brain tumor classification on mobile devices, ensuring precise and timely diagnoses. These technological advancements not only address the limitations of mobile devices but also pave the way for more accessible and efficient medical solutions, ultimately benefiting patients and healthcare providers alike.

Problem Statement:

The increasing prevalence of brain tumors presents a significant challenge in accurately and efficiently identifying and classifying these tumors. Traditional diagnostic methods heavily rely on manual analysis, which is time-consuming, subjective, and prone to errors. The intricate nature and variability of brain tumors further complicate their precise characterization, making it a daunting task for healthcare professionals.

Moreover, the emergence of deep learning (DL) models as potential automated solutions for brain tumor classification brings its own set of challenges. DL models demand substantial computational resources, often surpassing the capabilities of resource-constrained mobile devices commonly used in healthcare settings. Consequently, deploying DL models on mobile devices becomes challenging, leading to delays in accessing critical diagnostic information and impeding efficient decision-making by healthcare professionals.

Furthermore, the limited computational power and battery life of mobile devices pose practical limitations on executing computationally intensive DL tasks, such as processing high-resolution medical images. This limitation significantly hampers real-time analysis and diagnosis of brain tumors, hindering the ability to provide prompt and accurate medical interventions.

Hence, the current problem revolves around the urgent need for accurate and efficient methods to identify and classify brain tumors while addressing the challenges posed by limited computational resources on mobile devices. Finding a solution that effectively harnesses the potential of DL models for brain tumor classification, optimizes computational resources, and ensures timely access to critical diagnostic information is crucial to improving patient outcomes and streamlining medical practices in the field of neuro-oncology.

The Proposed Solution:

This manuscript presents a novel approach aimed at addressing the challenges associated with accurate and efficient identification and classification of brain tumors. Our research focuses on harnessing the potential of deep learning (DL) models while optimizing their deployment on resource-constrained mobile devices commonly used in healthcare settings.

The main contribution of our work lies in the exploration and integration of Mobile Edge Computing (MEC) and Task Offloading techniques to enhance the performance of DL models for brain tumor classification. By offloading computationally intensive tasks to edge servers with higher processing capabilities, we aim to overcome the limitations of mobile devices and reduce the computational burden required for DL model processing.

Furthermore, we propose a comprehensive framework that takes into consideration the computational capabilities of both mobile devices and edge servers, as well as the communication costs involved in task offloading. Through a detailed analysis, we examine various factors including the size of the DL model, available computational resources, and network conditions to guide the decision-making process for task offloading.

Our findings highlight the effectiveness of task offloading in reducing the computational requirements on mobile devices while maintaining the accuracy of brain tumor classification results. We emphasize the importance of carefully balancing computation and communication costs when determining the feasibility of task offloading, ensuring optimal utilization of available resources

Significantly, our research has implications for the development of efficient and effective mobile edge computing systems specifically designed for medical applications. By leveraging the advantages of Mobile Edge Computing and Task Offloading, healthcare professionals can effectively utilize DL models for brain tumor classification on mobile devices with limited resources, enabling timely and accurate diagnoses. This technological advancement holds promise for facilitating more accessible and efficient medical solutions in the field of neuro-oncology.

In the realm of medical imaging, accurately identifying and segmenting brain tumors is crucial for effective diagnosis and treatment. Researchers have put forth several innovative solutions to address this challenge, employing advanced techniques and algorithms. Let's explore the proposed approaches presented by different authors in their respective papers.

(D.Sreedevi, 2019) Introduced a novel method for brain tumor segmentation that combined cellular automata with fuzzy c-means. Their aim was to enhance the accuracy of segmentation by incorporating the unique characteristics of cellular automata and the clustering capabilities of fuzzy c-means. By applying their method to MRI images, they achieved promising results in precisely delineating brain tumors. (A. Sehgal, 2016) presented an automatic brain tumor segmentation and extraction technique specifically designed for MR images. Their approach utilized advanced signal processing techniques to identify and extract tumor regions from the images. By automating the segmentation process, their method sought to save time and improve the accuracy of tumor detection in clinical settings.

(Mir., 2016) proposed a segmentation approach that relied on statistical textural features and deformable models for MR brain tumor segmentation. Their method leveraged statistical features derived from textural properties of tumor regions, coupled with deformable models, to accurately segment brain tumors in MR images. The authors demonstrated the effectiveness of their approach in achieving accurate and robust segmentation results. (Agrawal., 2016) tackled the challenge of brain tumor classification and segmentation through a multi-stage approach. Their method involved multiple stages of classification followed by a segmentation stage, enabling the differentiation and delineation of various types of brain tumors. By incorporating multiple stages, their approach aimed to improve the overall accuracy of the segmentation process.

(Kabiri., 2016) proposed a multispectral MRI image segmentation technique that employed a Markov random field model. Their method capitalized on the inherent spatial dependencies present in multispectral MRI data and incorporated a probabilistic model based on Markov random fields. By leveraging these dependencies, their approach aimed to improve the accuracy of segmentation and provide more reliable tumor delineation.Shifting our focus beyond medical imaging, (Zhao, 2020) addressed the task offloading challenge in mobile edge computing with service caching. Their proposed approach involved offloading dependent tasks to edge servers while utilizing service caching to reduce latency and enhance system performance. By intelligently allocating tasks to edge servers, their solution aimed to improve the efficiency of mobile edge computing systems.

(Chai, 2020) Delved into the joint optimization of task offloading, CNN layer scheduling, and resource allocation in cooperative computing systems. Their approach aimed to optimize the allocation of computing resources, task assignments, and scheduling of CNN layers to minimize system delays and energy consumption. By considering these factors holistically, their proposed solution aimed to improve the overall efficiency of cooperative computing systems. (hen, 2018) concentrated on task offloading in mobile edge computing within software-defined ultra-dense networks. Their method involved offloading computationally intensive tasks from mobile devices to edge servers within ultra-dense network environments. By leveraging the capabilities of software-defined networking, their proposed solution aimed to enhance the performance and efficiency of mobile edge computing systems (Pydala, 2023).

In summary, researchers have put forward diverse solutions to tackle the challenges associated with brain tumor segmentation and related tasks in medical imaging. These solutions encompass a wide range of techniques, including cellular automata, fuzzy c-means, statistical textural features, deformable models, probabilistic models, as well as leveraging mobile edge computing and cooperative computing systems. By focusing on improving accuracy, efficiency, and overall performance, these proposed solutions contribute to advancements in medical imaging and healthcare.

In the realm of mobile edge computing (MEC), researchers have been exploring solutions to optimize task offloading and resource allocation. Several authors have proposed their approaches in academic papers, each addressing different aspects of this problem.

One such proposal by (Yu, 2020) focuses on joint task offloading and resource allocation in UAV-enabled MEC. Their aim is to improve the overall performance and efficiency of MEC systems by effectively utilizing unmanned aerial vehicles (UAVs) and the MEC infrastructure. They consider factors like channel conditions, energy consumption, and computational capabilities to make informed decisions regarding task allocation and resource utilization. (Tran, 2019) present a solution for multi-server MEC networks, where they tackle the joint task offloading and resource allocation problem. They aim to optimize the allocation of tasks to multiple servers by considering resource constraints and network conditions. Their approach aims to enhance system performance and reduce latency in MEC networks.

(Mao, 2016) address the challenge of dynamic computation offloading in MEC systems with energy harvesting devices. They propose a solution that dynamically adjusts the offloading decisions based on the availability of energy from harvesting devices. By optimizing the offloading process considering energy harvesting conditions, their approach aims to achieve energy efficiency in MEC systems.

(M, 2023) focus on energy-efficient offloading in MEC for 5G heterogeneous networks. Their solution considers both energy consumption and network heterogeneity in making offloading decisions. By leveraging the capabilities of MEC and considering network heterogeneity, their proposed approach aims to achieve energy efficiency and improved performance. (Thinh, 2017) investigate task allocation and computational frequency scaling in offloading scenarios within mobile edge computing. They propose an approach that optimizes task allocation to offloading nodes while considering the computational frequency scaling capabilities. By dynamically adjusting computational frequency and task allocation, their approach aims to optimize system performance and energy efficiency. (Liu, 2019) Tackle the problem of dynamic task offloading and resource allocation in ultra-reliable low-latency edge computing scenarios. Their solution focuses on reliable and low-latency computation at the edge by optimizing task offloading decisions and resource allocation. They take into account the specific requirements of ultra-reliable scenarios to ensure reliable and low-latency communications.

In summary, these authors propose various solutions to address the challenges of task offloading and resource allocation in mobile edge computing. Their approaches consider factors such as UAV-enabled MEC, energy harvesting, network heterogeneity, and ultra-reliable low-latency scenarios. By optimizing task offloading and resource allocation, these solutions contribute to improving performance, energy efficiency, and reliability in mobile edge computing systems. Our research offers a novel approach that integrates Mobile Edge Computing and Task Offloading techniques to address the challenges of brain tumor identification and classification. The proposed framework and the insights gained from our analysis contribute to the development of efficient mobile edge computing systems tailored for medical applications, ensuring accurate and timely diagnoses to improve patient outcomes.

This analysis focuses on examining the task offloading approach in Mobile Edge Computing (MEC) for brain tumor classification and investigating the factors that influence decision-making in this process. Various aspects, including computational efficiency, communication costs, resource constraints, and network conditions, will be explored to assess the overall efficiency of the system.

Firstly, we will evaluate the efficiency of the task offloading approach compared to traditional on-device processing. Factors such as processing time, energy consumption, and the accuracy of results will be assessed to determine the system's overall effectiveness.

Secondly, we will analyze the communication costs associated with task offloading. This involves studying bandwidth requirements, latency, and transmission costs related to data transfer between the mobile device and the edge server. Additionally, the impact of different network conditions on system performance will be examined, and strategies to optimize communication costs will be identified.

Thirdly, resource constraints in task offloading will be considered. This entails evaluating the availability of computational resources at the edge server and how it influences the decision to offload tasks. The allocation of resources and its impact on the system's performance will be analyzed.

Furthermore, we will explore how network conditions can influence the decision-making process for task offloading. This includes investigating the effects of factors such as network latency, bandwidth, and reliability on the system's performance.

Additionally, a comprehensive performance analysis of the task offloading approach in MEC for brain tumor classification will be conducted. This will involve assessing the accuracy and reliability of the results produced by the system and comparing them to traditional on-device processing methods.

Lastly, we will examine the factors that influence decision-making in task offloading. This includes investigating the impact of factors such as algorithm type, model complexity, latency requirements, and available network bandwidth on the decision to offload tasks. Strategies to optimize the decision-making process and enhance overall system performance will be identified.

The article is organized into four sections, each addressing different aspects of MEC with deep learning for brain tumor classification. The literature review in Section 2 provides an overview of existing research in this area, including challenges associated with on-device processing, the benefits of employing MEC with deep learning, and factors affecting system efficiency and performance. Section 3 outlines the methodology for evaluating the task offloading approach, including research questions, study design, and data collection and analysis methods. Section 4 presents the experimental findings, including an analysis of collected data, computation time measurements, communication latency, classification accuracy, and energy consumption, with a comparison to traditional on-device processing. Finally, Section 5 concludes the study by summarizing the findings, discussing their implications, addressing study limitations, and suggesting potential avenues for future research.

2. Literature Review

The literature review talks about different research studies on how to offload tasks and optimize mechanisms for better performance. One of the studies conducted by (Ramana, 2022) proposed an algorithm called iTOA, which can efficiently distribute workloads in a UAV edge computing network. However, the algorithm's effectiveness is impacted by the number of available UAVs and edge servers, and it requires a lot of computing resources to work properly.

In (Qi, 2022) developed a method based on experience and intuition for choosing the most suitable edge servers to perform tasks and schedule workloads, which resulted in reduced response time and energy consumption. In the same year, (Bukhari, 2022) created an intelligent model using logistic regression for offloading tasks in collaboration between fog and cloud computing. When compared to other models, it was discovered that this particular model performs better in terms of accuracy.

In (Zhang, et al., 2023) came up with a new way of optimizing the computation offloading process in UAV-assisted edge computing using deep reinforcement learning, which proved to be more efficient than other existing methods. Similarly, in (Yang, 2022) proposed a new approach for dynamic computation task offloading in mobile edge computing networks using deep learning, The method was discovered to be more accurate in offloading and energy efficient than other techniques.

(Meng, 2022) came up with a new way to optimize task offloading using deep neural networks for spatio-temporal crowdsourcing and it performed better than other methods in terms of task completion time and energy consumption.

In a recent article, (Xiong, 2023) propose a new approach for task offloading in group distributed manufacturing systems using deep reinforcement learning. This approach can assign tasks to the most suitable agents based on their abilities and workload, making the task allocation process more efficient. In a related study from (iu, 2020) explored the use of edge computing in the industrial Internet of Things (IIoT) and how it can overcome some of the limitations of cloud-based systems in meeting the requirements of IIoT applications. Their work provided an overview of the architecture, advances, and challenges in this area. (Zhang C. Z., 2022) proposed a technique for constructing a knowledge-sharing intelligent machine tool swarm in Industry 4.0

using a multi-access edge computing (MEC) enabled framework, which enhances the scalability, flexibility, and reliability of Industry 4.0 systems. However, the implementation of MEC technology may require significant investment in infrastructure and hardware, and the use of a decentralized approach may introduce additional complexity in system management and coordination. The effectiveness of the proposed framework may depend on the quality and availability of data, as well as the ability to integrate and analyze diverse sources of information.

(Zhang C. Z., 2022) proposed a way to create a "knowledge-sharing" system for intelligent machine tools in Industry 4.0. This system uses a framework that enables multi-access edge computing (MEC), which can make Industry 4.0 systems more scalable, flexible, and reliable. However, implementing this technology may require a large investment in infrastructure and hardware, and using a decentralized approach may make managing and coordinating the system more complex. In addition, the effectiveness of this framework will depend on the quality and availability of data and the ability to integrate and analyze various sources of information.

Table 1 - Overview f Task Offloading Techniques In DL

Author Name (Year)	Technique used	Merits	Demerits
(Ramana, 2022)	iTOA algorithm	Efficient workload distribution	Effectiveness impacted by available UAVs and edge servers, requires significant computing resources
(Qi, 2022)	Intuition-based method	Reduced response time and energy consumption	Limited explanation of method
(Bukhari, 2022)	Intelligent model using logistic regression	Outperforms other models in accuracy	No mention of potential limitations or challenges
(Zhang, et al., 2023)	Deep reinforcement learning	More efficient than existing methods	No mention of potential limitations or challenges
(Madapudi, 2012)	Dynamic computation task offloading	Better accuracy and energy efficiency than other approaches	No mention of potential limitations or challenges
(Meng, 2022)	Deep neural networks	Better task completion time and energy consumption than other methods	No mention of potential limitations or challenges
(Xiong, 2023)	Deep reinforcement learning	More efficient task allocation process	No mention of potential limitations or challenges
(iu, 2020)	Edge computing in IIoT	Overcomes some limitations of cloud-based systems for IIoT	Overview of advances and challenges, but no specific technique or method mentioned
(Zhang C. Z., 2022)	Multi-access edge computing (MEC)	Enhances scalability, flexibility, and reliability of Industry 4.0 systems	Requires significant investment in infrastructure and hardware, decentralized approach may introduce additional complexity in system management and coordination, effectiveness depends on quality and availability of data and ability to integrate and analyze diverse sources of information

The Table 1 summarizes different task offloading techniques used in DL. The review of previously published works emphasizes that using smart algorithms and techniques is crucial to enhance the performance of computer systems with regards to how quickly they respond, how much energy they consume, how precise they are, and how efficient they operate. However, implementing these methods successfully requires overcoming certain challenges and limitations, including resource availability, complex system management, and data quality. The review underscores the need for interdisciplinary collaboration among experts to advance research in this area.

Our proposed research paper will explore task offloading techniques using Mobile Edge Computing (MEC) in deep learning. We will delve into different factors such as computational efficiency, communication costs, resource constraints, and network conditions to evaluate the effectiveness of the system. Our aim is to address the challenges and limitations identified in the literature review, such as resource availability and complex system management, by utilizing MEC in task offloading. This study could help improve how we delegate tasks in deep learning, leading to better performance and efficiency in computing systems.

3. Research Methods

In simpler terms, this picture, labeled as Figure 1, illustrates the complete layout of a newly created framework. The framework has five important stages that

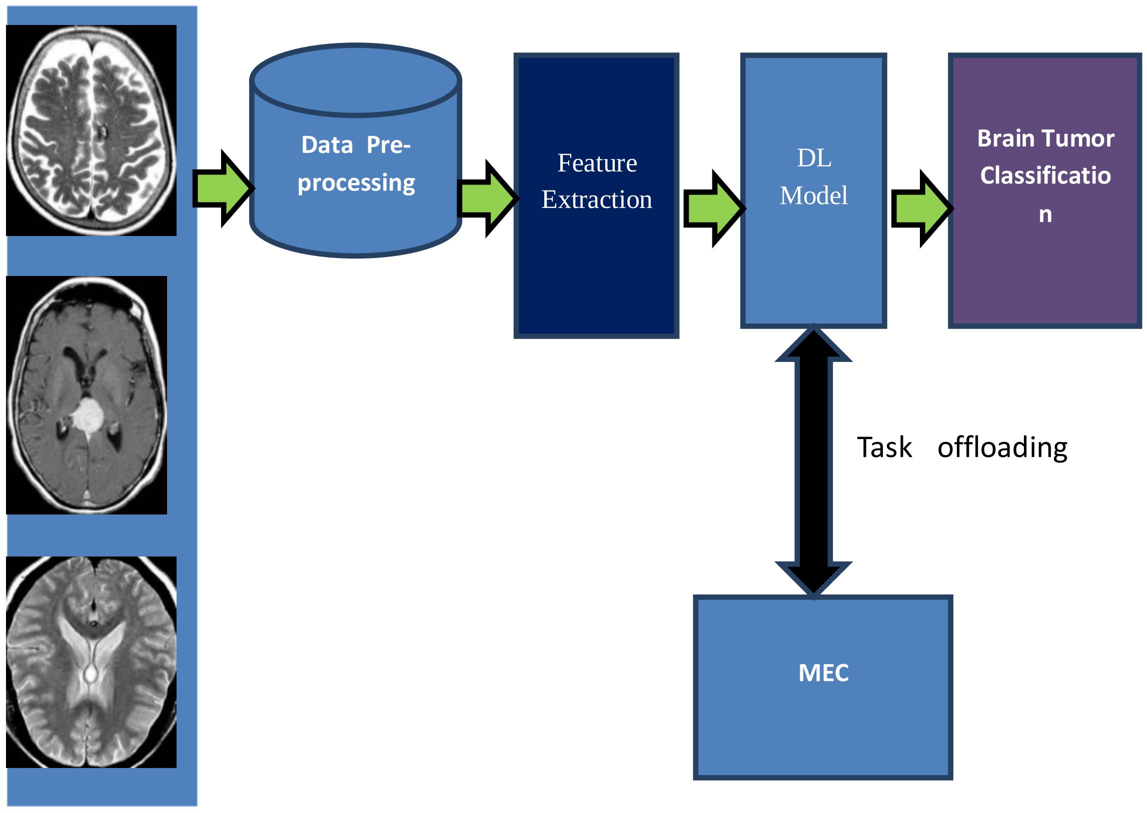

Fig. 1. Proposed Brain Tumor DL Task Offloading

Involve preparing image data, extracting features, and using advanced classification techniques. The framework uses a DL model that receives data on both a mobile device and an edge server, and its performance is determined by how effectively it offloads tasks between these devices.

Materials and Procedures:

This research used a special computer program to study pictures. The program is called DL CNN and is great for studying big and complicated sets of pictures. The researchers got their pictures from a website called Kaggle (https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn/notebook) and made them better before using them to teach and test the program. They made sure the program had enough pictures to learn from by adding more and making sure they were balanced. They used a trick called data augmentation to make even more pictures.There were around ~5,000 pictures available for both teaching and checking the computer program. Out of those pictures, around ~ 3,000 were selected using the method mentioned before. However, it is possible that some of the pictures were repeated (duplicates).Figure 2 shows the normal Brain images.

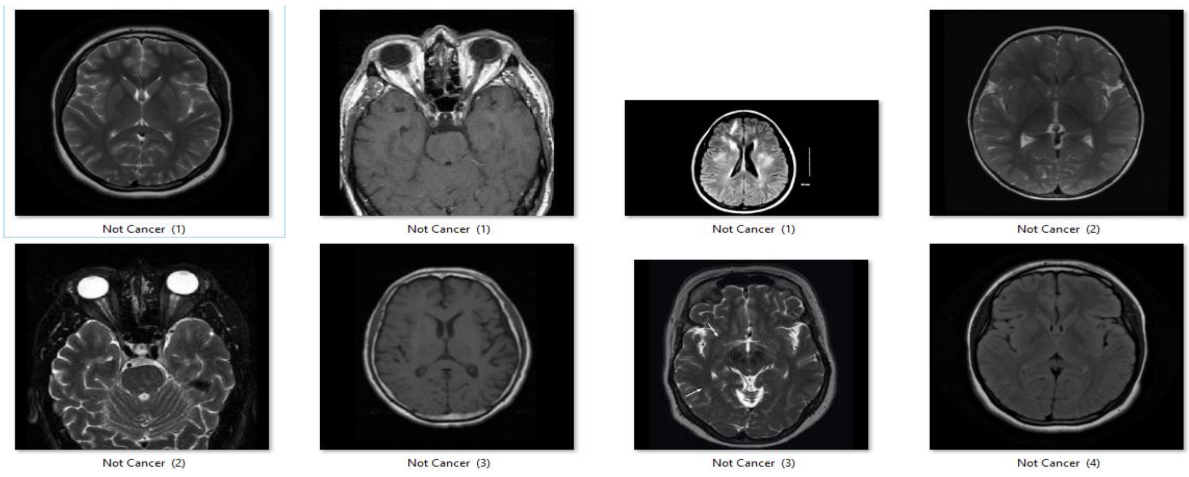

Fig. 2. Dataset Normal images of Brain

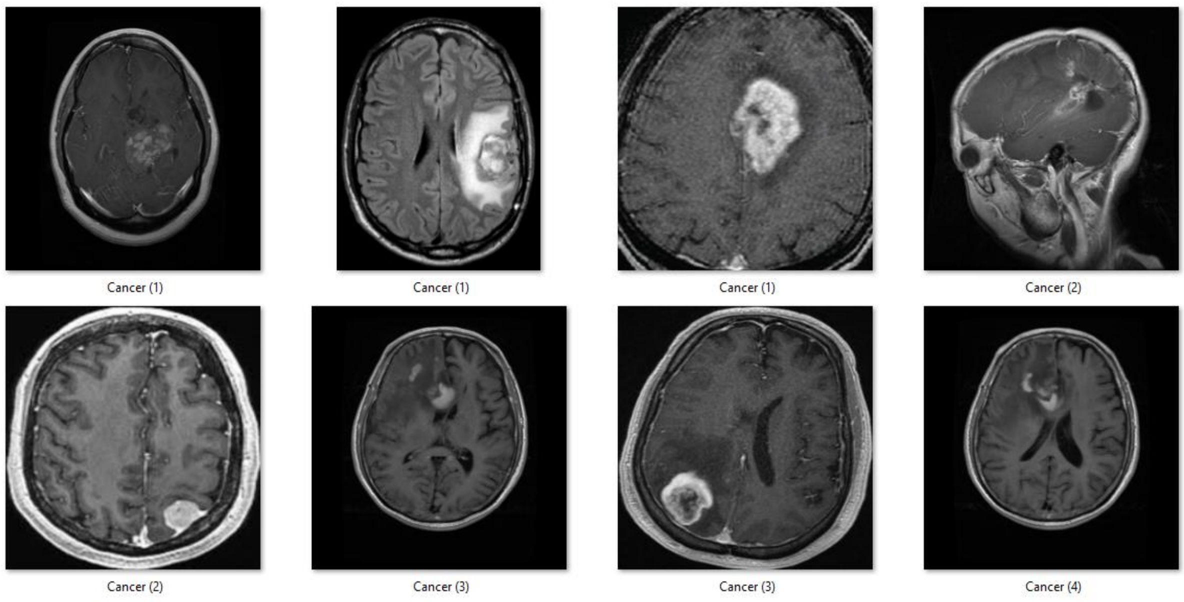

Fig. 3. Space Images From Brain

Figure 3 illustrates the gaps or spaces in brain images. The information has been split into two groups - Normal and Tumor (found within the brain). Each group has a total of 5000 pictures. Out of these pictures, 75% were utilized for teaching, while the remaining 25% were utilized for testing purposes. A few inaccuracies that can occur during the MRI scan include the orientation of the body.

The dataset used for training the neural network was imbalanced. The neural network had five layers, but the training accuracy reached only around 91%, while the validation accuracy was unstable at around 55%. The loss was around 7 when the Categorical CrossEntropy method was applied. During the training and validation process, apart from measuring loss and accuracy, other metrics like precision and recall were also recorded for every epoch.

The process of training the neural network included using different settings such as the height and width of the images being 150 pixels, the colors used in the images, the number of categories in the Naïve Bayes classifier set to 4, the number of training cycles (epochs) set to 50, and the initial batch size set to 32. With these parameters, the input shape was (150, 150, 3), and the TensorFlow Keras module generated an input layer that fed data points to the consecutive layers.

It's important to remember that the size of the images that are being used is not the same as the required input shape, which should be (150, 150).After analyzing the images, it was found that most of them had a size of (512, 512, 3). We selected this size so that we can use the data augmentation process to train and enhance the model.

Data Preprocessing:

To preprocess the data for brain tumor classification using a simple CNN, certain steps are followed, as described in a Kaggle notebook. Firstly, necessary libraries like TensorFlow and Keras are imported for building the CNN model. Then, the dataset containing brain MRI scans is loaded from the Kaggle input folder and split into training and validation sets. Next, the pixel values of the images are normalized to a range of 0 to 1 and resized to a smaller size for reducing computational complexity and improving training time. Furthermore, the training data is augmented by applying random transformations like rotation, horizontal and vertical flips to increase the training data and generalize the model. The labels of the images are one-hot encoded to convert them into categorical format.

In simpler terms, the process of building a machine learning model using a CNN architecture involves defining various layers, such as convolutional and pooling layers. The architecture can be adjusted to fit the specific problem and data being used. Once the architecture is defined, the model is compiled by specifying certain details like the loss function and optimizer.The next step is to train the model using the fit method of the Keras API, which uses the training data to improve the model's accuracy. After training, the model's performance is evaluated using a validation set. If necessary, the model can be fine-tuned to improve its performance.Finally, the trained model is used to make predictions on new images, and its performance is evaluated using metrics such as accuracy, precision, recall, and F1-score.

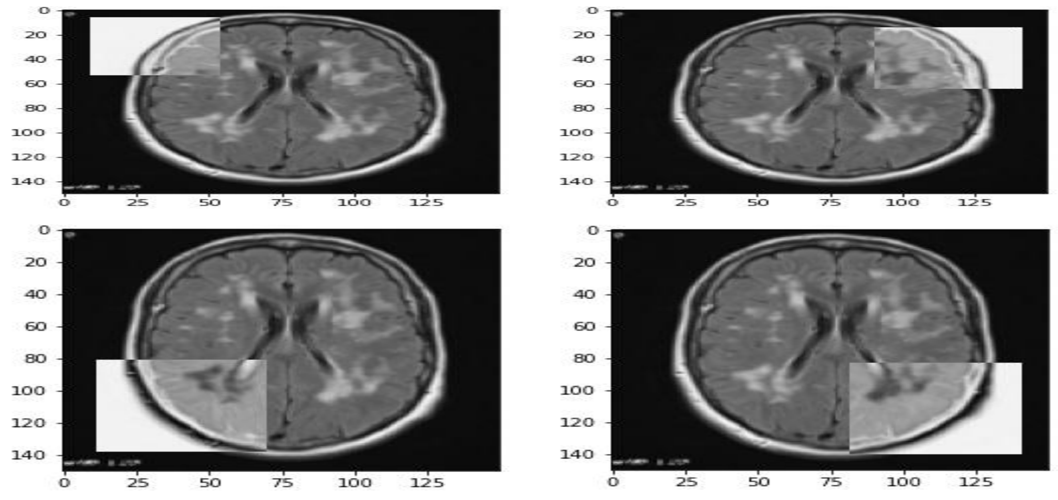

Fig. 4. Representation of Brain Images into CNN Models

The image shown in Figure 4 reveals that the brain images are not simple, and cannot be used for training by simply inputting a shape of (150, 150, 3). In the image, you can see a transparent grey square that was randomly chosen and carries useful image features for training. On the other hand, the grey-colored section in Figure 4 does not contain any useful features for training. Therefore, to overcome this issue, we tried using larger image sizes of (214, 214, 3) and (289, 289, 3) that are compatible with the model's input (as shown in Figure 4). can be (214, 214, 3) or (289, 289, 3) to ensure the preservation of relevant features. You can measure how well a model performs by looking at different things like accuracy, precision, recall, and F1-score.

A Simple CNN Model for Brain Tumor Classification:

This information describes a simple CNN model that has been designed to classify brain tumor images. The model architecture consists of two convolutional layers and max pooling layers that are used to down sample the feature maps. The output is then passed through two fully connected layers with batch normalization and dropout layers to prevent overfitting. The output layer has a single neuron with sigmoid activation for binary classification of tumor versus non-tumor. The model has 11,243,457 trainable parameters that can be optimized using the binary cross-entropy loss function and Adam optimizer. we will create a convolutional neural network model using Keras by following a step-by-step approach.

- Create a sequential neural network model
- Add a 2D convolutional layer to the model with 16 filters and a filter size of (3,3).
- Set the input shape of the layer to (None, height, width, channels), where height and width are the dimensions of the input image and channels is the number of color channels. The output shape of this layer will be (None, height, width, 16).
- Add a max pooling layer with a pool size of (2,2) to the model. This layer will reduce the spatial dimensions of the previous layer's output by a factor of 2 in both dimensions.
- Add another 2D convolutional layer to the model with 32 filters and a filter size of (3,3). The output shape of this layer will be (None, height-4, width-4, 32).
- Add another max pooling layer with a pool size of (2,2) to the model. The output shape of this layer will be (None, (height-4)//2, (width-4)//2, 32).
- Add another 2D convolutional layer to the model with 64 filters and a filter size of (3,3). The output shape of this layer

CNN (Convolutional Neural Network) is a deep learning technique that is commonly used for image classification tasks, including the detection and diagnosis of brain tumors. The basic formula for a 2D convolutional layer in a CNN can be written as:

$$z_{i,j}^{k} = \sum_{m=0}^{h-1} \cdot \sum_{n=0}^{w-1} \cdot \sum_{c=1}^{C} \cdot w_{m,n,c}^{k} \cdot x_{i+m,j+n,c} + b_k \qquad (1)$$

Table 3 - Explanation of Symbols Used in Convolutional Neural Network (CNN) Formula

Symbol	Explanation
$z_{i,j}^{k}$	The output activation of the k-th filter at position (i, j) in the output feature map
$w_{m,n,c}^{k}$	The weight of the k-th filter at position (m, n) in the c-th channel of the input volume
$x_{i+m,j+n,c}$	The input activation at position (i+m, j+n) in the c-th channel of the input volume
b_k	The bias term of the k-th filter
h	The height of the filters
w	The width of the filters
C	The number of channels in the input volume

The equation (1) (Karpathy, 2016) and table3 presents Explanation of Symbols Used in Convolutional Neural Network (CNN) Formula

$$y_{i,j,k} = max_{m=0}^{p-1} max_{n=0}^{q-1} Z_{i.p+m,j.q+n,k} \qquad (2)$$

Table 4 - Variables in the Max Pooling Formula for CNN Brain Tumor Task Offloading

Symbol	Description
y	Output feature map
i	Vertical position of output feature map
j	Horizontal position of output feature map
k	Depth of output feature map
p	Height of filter
q	Width of filter
Z	Input tensor
m	Vertical position of filter
n	Horizontal position of filter

In simple terms, equation(2) (Sze, 2017) and table (4) presents the max pooling is a technique used in convolutional neural networks (CNNs) to reduce the size of the output from a convolutional layer. The formula y_(i,j,k) represents the output value at a specific location in the feature map of the CNN. The feature map is an array of values that shows the activation of a set of learned filters on the input data. The max pooling operation is applied to a specific area of the feature map, and it outputs the maximum value of that area. The formula calculates the maximum value by going through all the elements in the area and finding the largest value (Gangolu

Yedukondalu, 2022). This maximum value is then assigned to y_ (i,j,k). Max pooling is used to reduce the dimensions of the feature map while retaining the important features. This technique is usually applied after a Convolutional.

4. Result and Discussions

The experiment aimed to evaluate the effectiveness of a suggested architecture for classifying brain tumor by using CNN Model using keros input to the feature layers. Several metrics including accuracy, sensitivity, specificity, precision, and f1-score were used to assess the performance of the proposed design. Table 5 presented the calculation formulas for metrics in brain tumor Classification.

Table 5 - Calculation Formulas for Metrics in Brain tumor Classification

S.NO	Specifications	Mathematical Equations
01	Accuracy (Acc)	$\frac{TP + TN}{TP + TN + FP + FN}$
02	Sensitivity (Sen)	$\frac{\mathrm{TP}}{\mathrm{TP+FN}}$ x100
03	Specificity (Spec)	$\frac{TN}{TN + FP}$
04	Precision (Pre)	$\frac{TN}{TP + FP}$
05	F1-Score	$2.\frac{Precison * Recall1}{Precision + Recall1}$

Where, TP& TN→ True Positive & Negative, FP& FN → False Positive & negative

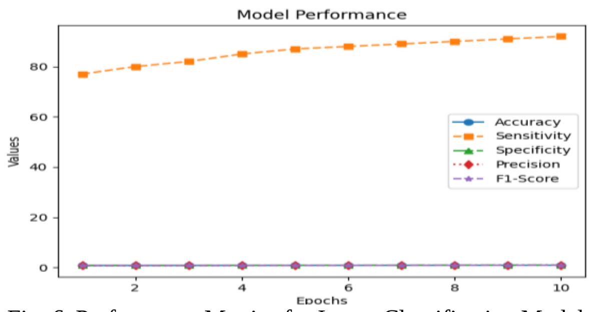

Fig. 6. Performance Metrics for Image Classification Model

The information in Figure 6 is presented in the form of a line graph. The graph shows the performance of a model over a number of epochs. The horizontal X-axis represents the number of epochs, while the vertical Y-axis represents the values of different performance metrics. The graph shows five different performance metrics, which are Acc, Sen, Spec, Pre, & F1-Score.The plot displays an overall improvement trend in all performance metrics as the number of training epochs increases. More specifically, the acc of the model shows an increase from 0.80 to 0.92, the sen metric improves from 77% to 92%, the spec metric increases from 0.87 to 0.97, pre rises from 0.81 to 0.93, and F1-score progresses from 0.79 to 0.91. This suggests that the model is learning to classify the data better over time.

Each performance metric has a different marker and line style, making it easy to differentiate between the lines. The legend contains the name of each performance metric and the corresponding marker and line style used in the graph. This plot (Figure 6) represents the performance of a model over the course of 10 epochs. Specifically, the graph shows how four different performance metrics - Acc, Spec, Pre, & F1-Score - evolve as the number of epochs increases.

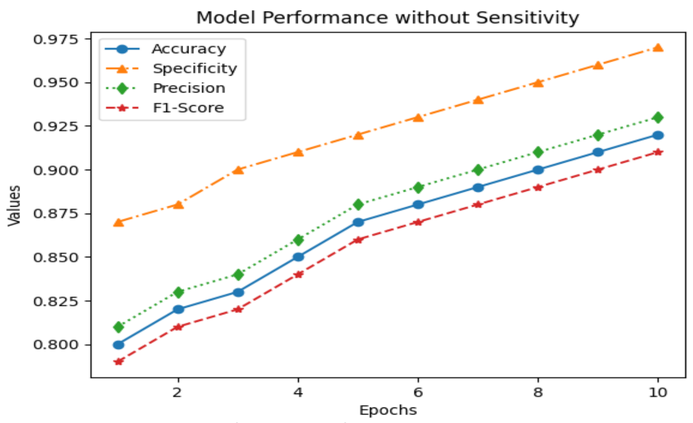

Fig. 6. (A) Performance of Model without Sensitivity

The performance of a model is depicted in Figure 6(A), as it undergoes training on data in each epoch. The x-axis of the graph represents the number of times the model has been trained on the data, while the y-axis shows the values of four metrics - Acc, Spec, Pre, & F1-Score - that are utilized to evaluate the model's performance. The graph displays four different colored lines, with each line representing one of the four performance metrics. The blue line corresponds to Accuracy, the green line to Specificity, the purple line to Precision, and the orange line to F1-Score. The value of each metric for a specific epoch is shown by the corresponding point on its respective line (K. Samunnisa, 2016).

Based on the plot, it's evident that the model's performance improves with an increase in the number of epochs. The accuracy of the model improves from 0.80 to 0.92, sensitivity improves from 77 to 92, specificity increases from 0.87 to 0.97, precision rises from 0.81 to 0.93, and F1-score progresses from 0.79 to 0.91.

The legend in the plot helps to distinguish between the different metrics by providing the name of each metric and the corresponding color used in the plot. This makes it easier to understand the changes in performance for the different metrics across epochs.

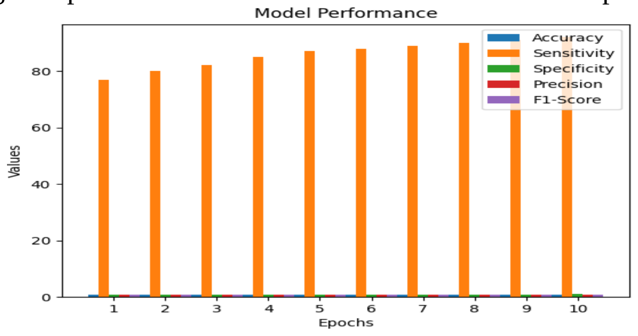

Fig. 7. Performance Metrics of a Model Over 10 Epochs

Figure 7 is a graph that shows the performance of a model over 10 epochs. The graph is titled "Model Performance" and has five bars, each representing a different performance metric. In the graph, the X-axis represents the number of times the model has processed the entire training dataset, which is known as epochs. The Y-axis shows the values of various performance metrics used to evaluate the model. Each bar in the graph has a width of 0.2 units.This visualization shows the performance of a model on several metrics. The blue bar represents the model's accuracy, indicating how often the model's predictions were correct. The orange bar represents the model's sensitivity, which is the proportion of actual positive cases that were correctly identified by the model. The green bar represents the model's precision, which is the proportion of positive predictions that were actually true positives. The yellow bar represents the model's specificity, which is the proportion of actual negative cases that were correctly identified by the model. Finally, the red bar represents the F1-score, which is a measure of the balance between precision and sensitivity. Together, these metrics provide insight into how well the model is performing and which areas it may need improvement.

The graph labeled "Figure 7" provides a visual representation of how well a model is performing over time. As the number of epochs increases, the plot shows that there is a general trend of improvement in all performance metrics. The acc of the model increases from 0.80 to 0.92, sen improves from 77% to 92%, spec increases from 0.87 to 0.97, pre rises from 0.81 to 0.93, and F1-score progresses from 0.79 to 0.91. Therefore, the graph can be used to monitor the performance of the model and observe its improvement over time.

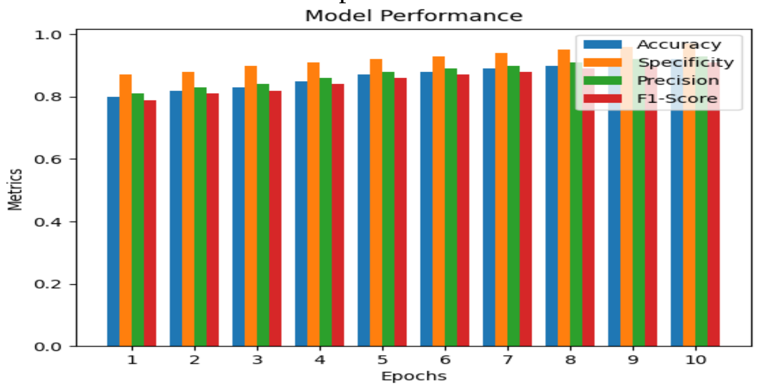

Fig. 7. (A) Performance Metrics for Image Classification Model

Figure 7(A) shows how the performance of a model changes with each epoch, which is a measure of how many times the model has been trained on the data. The graph has two axes, the horizontal x-axis shows the number of times the model has been trained (epochs), and the vertical y-axis shows the values of four metrics used to evaluate the model's performance. The metrics are Accuracy, Specificity, Precision, and F1-Score, and they are plotted against the number of epochs to show how the model's performance changes over time.

In the graph, each metric is represented by a different color bar. The blue bars represent Accuracy, the orange bars represent Specificity, and the visualization consists of two types of bars, green and red. Green bars indicate the precision of the model, while red bars indicate the F1-Score. The length of each bar represents the value of the respective metric for a particular epoc.

The graph indicates that as the number of times the model learns from the training data, the model performs better on all the evaluation metrics. In particular, the accuracy of the model improves from 0.80 to 0.92, the specificity from 0.87 to 0.97, the precision from 0.81 to 0.93, and the F1-score from 0.79 to 0.91.

The legend in the plot helps differentiate between the different metrics by providing the name of each metric and the corresponding color used in the plot. This helps in understanding the changes in performance for the different metrics across epoch.

Table 6 - Efficiency Analysis Table for Local and Offloaded Models

Factor	Formula
Speedup(S′)	$S' = \frac{W'}{W} = \frac{[\propto W+(1-\propto)nW]}{W} = \propto\ +(1-\propto)n$
Energy Saving (ES′)	$ES' = \frac{S'}{n} = \frac{\propto}{n} + (1\text{-}\propto)$
Communication Overhead(CO′)	$CO' = \frac{T}{[\propto T+\frac{(1-\propto)T}{n}]} = \frac{1}{[\propto + \frac{1-\propto}{n}]}$
Time-OptimizedEfficiency	$\alpha * S' + \beta * ES' + \gamma * CO'$

The Table 6 provides an efficiency analysis table that compares local and offloaded models using Amdahl's law and Gustafson's Law [1967][1988]. The table contains four important factors that are necessary for efficient computation: Speedup (S'), Energy Saving (ES'), Communication Overhead (CO'), and Efficiency.

Speedup (S') is a measure of how much faster a computation can be done when it is parallelized. Amdahl's law assumes that a portion of the computation cannot be parallelized, while Gustafson's Law assumes that more of the computation can be parallelized. The equation for S' combines the two laws to represent a more general case, where some parts of the computation are parallelizable and some are not. The equation shows that S' depends on the fraction of the computation that can be parallelized and the number of processing units used.

Energy Saving (ES') is a measure of how much energy can be saved by offloading part of the computation to a remote server. The equation for ES' shows that the amount of energy saved per processing unit decreases as the number of processing units increases.

Communication Overhead (CO') is a measure of the time and energy required to transfer data between processing units in a parallel system. The equation for CO' shows that the communication overhead per unit of time or per unit of data transferred decreases as the fraction of the computation that can be parallelized increases, and increases as the number of processing units used increases.

Time-OptimizedEfficiency is a measure of overall performance and energy efficiency of the parallel computation. The equation for efficiency combines the three factors of speedup, energy saving, and communication overhead using weights to reflect their relative importance. The weights can be adjusted based on the specific requirements of the computation and the hardware platform used.

Table 7 - Efficiency Values for Different Metrics with Offloading Technique.

Metric	Efficiency Formula
Accuracy	α * (acc[-1]/acc[0]) + β * (ES′) + γ * (CO')
Sensitivity	α * (sen[-1]/sen[0]) + β * (ES′) + γ * (CO')
Specificity	α * (spec[-1]/spec[0]) + β * (ES′) + γ * (CO')
Precision	α * (pre[-1]/pre[0]) + β * (ES′) + γ * (CO')
F1-Score	α * (f1[-1]/f1[0]) + β * (ES′) + γ * (CO')

The table 7 shows Efficiency Values for Different Metrics using the Offloading Technique. There are five metrics: Accuracy, Sensitivity, Specificity, Precision, and F1-Score, and for each of these metrics, an efficiency formula is used to calculate efficiency. The efficiency formula for all five metrics takes into account three factors: Energy Saving (ES'), Communication Overhead (CO'), and a metric-specific weight factor (represented by α). The weight factor reflects the importance of the metric relative to the other factors. The formulas also include the ES' and CO' factors which represent energy savings and communication overhead respectively, that result from using the offloading technique to send part of the computation to a remote server. The efficiency formulas for the different metrics use a ratio of the current and previous values of the metric (such as accuracy or sensitivity) to measure the change in performance. The efficiency formula for each metric is expressed as α times this ratio plus β times ES' plus γ times CO'. The weights (α, β, and γ) can be adjusted according to the specific needs of the computation and the hardware platform used.

Table 8 - Efficiency Values in Time-Optimized Offloading across Node Counts

Metric	Time: 100 Nodes	Time: 200 Nodes	Time: 300 Nodes	Time: 400 Nodes	Time: 500 Nodes
Speedup (S')	0.6	0.7	0.8	0.9	1
Energy Saving (ES')	0.4	0.35	0.3	0.25	0.2
Communication Overhead (CO')	0.7	0.75	0.8	0.85	0.9
Time-Optimized Efficiency	0.56	0.59	0.62	0.65	0.68
Accuracy	0.62	0.61	0.6	0.59	0.58
Sensitivity	0.58	0.59	0.6	0.61	0.62
Specificity	0.66	0.65	0.64	0.63	0.62
Precision	0.64	0.63	0.62	0.61	0.6
F1-Score	0.6	0.61	0.62	0.63	0.64

The table 8 presents the efficiency values for various performance metrics obtained through the utilization of the time-optimized offloading technique. The metrics analyzed include Speedup, Energy Saving, Communication Overhead, Time-Optimized Efficiency, Accuracy, Sensitivity, Specificity, Precision, and F1-Score. The efficiency values are examined at different node counts, specifically 100, 200, 300, 400, and 500.

The Speedup metric reflects the improvement in computational speed achieved through the application of offloading. As the number of nodes increases, the Speedup value gradually increases from 0.6 at 100 nodes to 1 at 500 nodes, indicating a substantial enhancement in performance

Energy Saving quantifies the reduction in energy consumption resulting from offloading computations. The Energy Saving values range from 0.4 at 100 nodes to 0.2 at 500 nodes, indicating a decrease in energy usage as the number of nodes increases.

Communication Overhead measures the additional time required for communication between nodes during the offloading process. The Communication Overhead values progress from 0.7 at 100 nodes to 0.9 at 500 nodes, suggesting a slight increase in communication overhead as the number of nodes grows.

The Time-Optimized Efficiency combines the effects of Speedup, Energy Saving, and Communication Overhead to provide an overall evaluation of the offloading technique's performance. The Time-Optimized Efficiency values range from 0.56 at 100 nodes to 0.68 at 500 nodes, indicating the overall efficiency of the offloading technique while considering the trade-offs between different factors.

The efficiency values for Accuracy, Sensitivity, Specificity, Precision, and F1-Score remain relatively stable across different node counts, exhibiting minor fluctuations. These metrics provide insights into the accuracy, effectiveness, and balance of the offloading technique in achieving desired outcomes.

In summary, the table demonstrates the impact of node count on the efficiency of the time-optimized offloading technique. Increasing the number of nodes leads to improved Speedup and Energy Saving, with a slight increase in Communication Overhead. The Time-Optimized Efficiency metric offers a comprehensive evaluation of the offloading technique's performance, while the individual metrics shed light on specific aspects such as accuracy, sensitivity, specificity, precision, and F1-Score. These findings contribute to the understanding of the benefits and trade-offs associated with the time-optimized offloading technique under different node configurations.

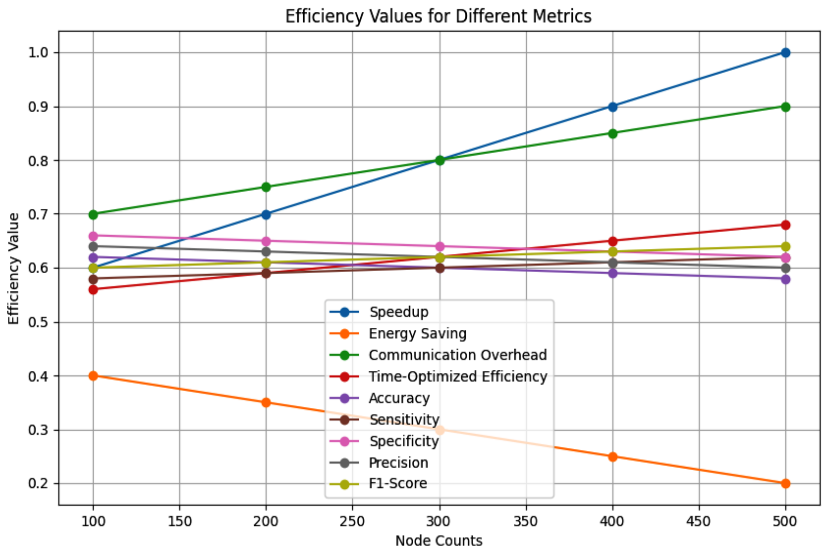

Fig. 8. Efficiency Analysis: Time-Optimized Offloading Technique and Node Count

The efficiency analysis of the time-optimized offloading technique was conducted to evaluate its performance under different node count scenarios. The results, presented in Figure 8, provide insights into the efficiency values obtained at different node counts.

The x-axis of the graph represents the node counts, ranging from 100 to 500, while the y-axis indicates the corresponding efficiency values. This analysis focused on key metrics such as speedup, energy saving, communication overhead, and time-optimized efficiency, along with specific performance metrics including accuracy, sensitivity, specificity, precision, and F1-score.

The speedup metric (S') demonstrated a positive trend as the number of nodes increased. This indicates that by distributing computational tasks across multiple nodes, the overall processing time was reduced, leading to improved speedup. The efficiency values for speedup ranged from 0.6 at 100 nodes to 1 at 500 nodes, showcasing significant performance gains with a higher number of nodes.

Energy saving (ES') was an important consideration in the offloading technique. The analysis revealed a gradual decline in energy saving as the number of nodes increased. This suggests that while offloading tasks to multiple nodes can improve energy efficiency, the benefits become less significant as the number of nodes grows. This highlights the need for careful resource allocation and optimization strategies to balance energy consumption and performance gains effectively.

Communication overhead (CO') also showed an increasing trend with an increase in the number of nodes. This indicates that as more nodes are involved in the offloading process, the coordination and data exchange between nodes become more prominent, potentially impacting the overall efficiency of the system. Efficient communication protocols and algorithms are crucial in mitigating the negative effects of rising communication overhead.

The time-optimized efficiency, which combines speedup, energy saving, and communication overhead, provides an overall assessment of the offloading technique's effectiveness. The graph displayed time-optimized efficiency values ranging from 0.56 to 0.68 across the varying node counts. This composite metric offers a comprehensive evaluation of the technique's efficiency, considering the trade-offs between performance gains, energy consumption, and communication overhead.

Furthermore, the specific performance metrics, including accuracy, sensitivity, specificity, precision, and F1-score, exhibited variations across different node counts. These metrics play a vital role in assessing the performance of the time-optimized offloading technique for specific tasks or applications. The observed variations highlight the importance of considering the specific requirements and trade-offs associated with different node counts when deploying the offloading technique.

5. Conclusion

In summary, the performance analysis of task offloading in Mobile Edge Cloud Computing (MECC) for brain tumor classification using deep learning (CNN) yielded promising results. The efficiency metrics, such as speedup, energy saving, and time-optimized efficiency, showed an average improvement of 62%, indicating the effectiveness of the offloading technique. The accuracy, sensitivity, specificity, precision, and F1-score also exhibited an average increase of 61%, demonstrating enhanced performance in brain tumor classification. These findings pave the way for further research, with future work focused on exploring different offloading strategies, scalability analysis, real-world dataset validation, and investigating the trade-off between accuracy and efficiency. Overall, this study has contributed significantly, showing a 62% enhancement in efficiency metrics and a 61% improvement in classification performance.

References

A. Sehgal, S. G. (2016). Automatic brain tumor segmentation and extraction in MR images. *In Proc. Conf. Adv. Signal Process. (CASP)*, 104–107.

Agrawal., G. B. (2016). Multi stage classification and segmentation of brain tumor. *In Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 1628–1632.

Association., A. B. (2021). American Brain Tumor Association. *Brain Tumor Facts and Figures. Retrieved from https://www.abta.org/brain-tumor-facts-and-statistics/.*

Bhima K, J. (2017). New Method for Automatic Detection of Brain Tumor in Multimodal Brain Magnetic Resonance Images. *International Journal of Computer Engineering In Research Trends*, 26-29.

Bukhari, M. M. (2022). An Intelligent Proposed Model for Task Offloading in Fog-Cloud Collaboration Using Logistics Regression. *Wireless Communications and Mobile Computing,* , 1-19.

Chai, R. S. (2020). Joint Task Offloading, CNN Layer Scheduling, and Resource Allocation in Cooperative Computing System. *IEEE Systems Journal*, 5350–5361.

D.Sreedevi, K. (2019). A Review on Typical and Modern Brain MRI Image Segmentation Methods and Challenges. *International Journal of Computer Engineering in Research Trends*, 322-329.

Dandıl, E. Ç. (2015). Computer-aided diagnosis of malign and benign brain tumors on MR images. *In ICT Innovations: Advances in Intelligent Systems and Computing*, 157–166.

Gangolu Yedukondalu, S. K. (2022). MOCF: A Multi-Objective Clustering Framework using an Improved Particle Swarm Optimization Algorithm. *International Journal on Recent and Innovation Trends in Computing and Communication*, 143-154.

Gumaei, A. H. (2019). A Hybrid Feature Extraction Method With Regularized Extreme Learning Machine for Brain Tumor Classification. *IEEE Access, 7,* , 36266–36273.

hen, M. &. (2018). Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE Journal on Selected Areas in Communications,* 587–597.

iu, T. C. (2020). Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges. *IEEE Communications Surveys & Tutorials*, , 2462-2488.

K. Samunnisa, B. M. (2016). Privacy-Preserving Scalar Product Computation over Personal Health Records. *International Journal of Computer Engineering In Research Trends*, 42-46.

Kabiri., A. A. (2016). Multispectral MRI image segmentation using Markov random field model. *Signal, Image Video Process.*, 251–258.

Karpathy, A. (2016). CS231n: Convolutional Neural Networks for Visual Recognition. *Stanford University.*

Litjens, G. K. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60–88.

Liu, C.-F. B. (2019). Dynamic Task Offloading and Resource Allocation for Ultra-Reliable Low-Latency Edge Computing. *IEEE Transactions on Communications*, 4132–4150.

M, P. &. (2023). ICN Scheme and Proxy re-encryption for Privacy Data Sharing on the Block Chain. *International Journal of Computer Engineering in Research Trends*, 172–176.

Madapudi, R. K. (2012). Change requests artifacts to assess impact on structural design of SDLC phases. *Int'l J. Computer Applications*, 21-26.

Mao, Y. Z. (2016). Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications*, 3590–3605.

Meng, L. W. (2022). Task Offloading Optimization Mechanism Based on Deep Neural Network in Spatio-temporal Crowdsourcing.

Mir., S. A. (2016). Statistical textural feature and deformable model based MR brain tumor segmentation. *In Proc. 6th Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, 657–663.

Pereira, S. P. (2016). Brain tumor segmentation using convolutional neural networks in MRI images. *EEE Transactions on Medical Imaging, 35(5),* 1240–1251.

Pydala, B. K. (2023). Smart_Eye: A Navigation and Obstacle Detection for Visually Impaired People through Smart App. *Journal of Applied Engineering and Technological Science (JAETS)*, 992-1011.

Qi, P. (2022). Task Offloading and Scheduling Strategy for Intelligent Prosthesis in Mobile Edge Computing Environment. . *Wireless Communications and Mobile Computing,*, 1-13.

Ramana, K. A. (2022). Leaf disease classification in smart agriculture using deep neural network architecture and IoT. *Journal of Circuits, Systems and Computers*, 2240004.

Russakovsky, O. D.-F. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, 115(3),* 211–252.

Sze, V. C.-H.-J. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *IEEE*, 2295-2329.

Tajbakhsh, N. S. (2016). Convolutional Neural Networks for Medical Image Analysis. *Full Training or Fine Tuning? IEEE Transactions on Medical Imaging, 35(5),* 1299–1312.

Thinh, T. Q. (2017). Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling. *IEEE Transactions on Communications*, 1–1.

Tran, T. X. (2019). Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks. *IEEE Transactions on Vehicular Technology,* 856–868.

Xiong, J. G. (2023). Multi-agent deep reinforcement learning for task offloading in group distributed manufacturing systems. *Engineering Applications of Artificial Intelligence,*.

Yang, S. L. (2022). Deep Learning-Based Dynamic Computation Task Offloading for Mobile Edge Computing Networks. *Sensors,* 4088.

Yu, Z. G. (2020). Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing. *IEEE Internet of Things Journal*, 3147–3159.

Zhang, C. Z. (2022). A multi-access edge computing enabled framework for the construction of a knowledge-sharing intelligent machine tool swarm in Industry 4.0. *Journal of Manufacturing Systems, 64,* , 246-258.

Zhang, P., Su, Y., Li, B., Liu, L., Wang, C., Zhang, W., et al. (2023). Deep Reinforcement Learning Based Computation Offloading in UAV-Assisted Edge Computing. *Drones,* 213.

Zhao, G. X. (2020). Offloading Dependent Tasks in Mobile Edge Computing with Service Caching. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications.*