Journal of Information Systems Engineering and Management
2025, 10(29s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Evaluating Deep Learning Model Performance on Raspberry Pi 4 for COVID-19 Diagnosis

Mr.Bharat Tank[1], Dr.Mitul Patel[2], Dr.Khemraj Deshmukh[3],

[1]Phd Scholar, Parul University, Vadodara, Gujarat, India

[2]Assistant Professor, Parul University, Vadodara, Gujarat, India

[3]Assistant Professor, Parul University, Vadodara, Gujarat, India

Corresponding Author

kdeshmukh.phd2019.bme@nitrr.ac.in

ARTICLE INFO

Received: 30 Dec 2024

Revised: 05 Feb 2025

Accepted: 25 Feb 2025

ABSTRACT

The global coronavirus disease (COVID-19) pandemic has highlighted the urgent need for accessible, effective, and accurate diagnostic tools, especially in low- or no-service settings. Chest X-ray, a widely used diagnostic test, provides a rapid and non-invasive way to identify COVID-19 symptoms. Deploying deep learning models for COVID-19 detection on resource-constrained edge devices, such as the Raspberry Pi 4, requires a balance between model accuracy, inference speed, and hardware limitations. This study evaluates the performance of four deep learning models—ResNet18, ResNet50, DenseNet121, and SqueezeNet—based on key metrics such as inference time, memory usage, and model size after TensorFlow Lite conversion. Experimental results show that SqueezeNet offers the best trade-off, achieving the fastest inference time (10.76s per 100 images) and the lowest memory usage (2.8MB), making it the most suitable for real-time edge deployment. In contrast, ResNet50, despite its high accuracy, has the longest inference time (42.32s) and highest memory consumption (90MB), limiting its feasibility for Raspberry Pi-based applications. The findings highlight the importance of selecting lightweight architectures for efficient and scalable deep learning-based COVID-19 detection on edge devices. Detailed performance parameters including sensitivity, specificity, accuracy, reproducibility, and inference time were analysed to assess the suitability of each model for clinical trials. This study demonstrates the revolutionary potential of combining deep learning with portable devices to provide effective diagnostic tests for COVID-19 and similar respiratory diseases, address healthcare inequities, and facilitate timely interventions for epidemic control.

Keywords: Deep Learning, Raspberry Pi 4, Edge AI, COVID-19 Detection, Model Optimization, TinyML.

INTRODUCTION

The rapid spread of COVID-19 globally has highlighted the urgent need for easy-to-use, effective, and accurate diagnostic tools. Traditional diagnostic methods such as reverse transcription polymerase chain reaction (RT-PCR) are highly reliable but are often limited by logistical challenges, cost, and time delays. In this context, chest X-ray imaging appears to be a promising alternative due to its versatility, speed, and ability to detect abnormal lung involvement with COVID-19. However, the interpretation of these images requires specialized knowledge that is often unavailable in confined environments. A deep learning model for the automatic detection of COVID-19 on a low-cost portable platform (Raspberry Pi 4) is implemented and evaluated in this study to close this gap. The COVID-19 pandemic has posed significant challenges to global healthcare, necessitating the development of rapid and accurate diagnostic methods. Reverse transcription polymerase chain reaction (RT-PCR) is still the gold standard for testing, but it has some drawbacks like being expensive, taking a long time to complete, and requiring specialized laboratory facilities [9]. To detect COVID-19 from medical imaging data such as chest X-rays (CXR) and computed tomography (CT) scans, AI-driven diagnostic techniques, particularly deep learning (DL) models, have gained attention [2], [11]. Convolutional neural networks (CNNs) are widely used for automated feature extraction and classification [3, 6]. Deep learning-based approaches have demonstrated exceptional performance in medical image classification. CNN

architectures like ResNet, DenseNet, and MobileNet have been the subject of a number of investigations and have demonstrated high accuracy in distinguishing COVID-19 from other pulmonary infections [14, 13]. In addition, recent studies have highlighted the viability of putting AI models on edge devices like Raspberry Pi for portable and real-time diagnosis [1, 12].Despite the potential advantages, deploying deep learning models on edge computing devices presents challenges such as computational limitations, storage constraints, and inference speed [5], [8]. Several model compression and optimization techniques, including quantization and pruning, have been proposed to reduce model size while preserving diagnostic performance [4]. Comparative analyses of deep learning models for COVID-19 detection and object detection on edge devices have also been conducted to optimize performance and efficiency [7], [10].

This study focuses on developing and deploying optimized CNN models for COVID-19 detection from chest X-ray images on a Raspberry Pi 4 system. The research involves data preprocessing, transfer learning, model fine-tuning, and deployment using TensorFlow Lite. Four CNN architectures (ResNet18, ResNet50, DenseNet121, and SqueezeNet) will be evaluated based on key metrics, including accuracy, precision, recall, F1-score, and inference time. The findings of this study will contribute to the development of low-cost, real-time, and AI-driven COVID-19 detection solutions, particularly for resource-limited settings.

However, the use of computational models for devices with limited processing and memory has faced serious probl ems. Raspberry Pi 4 provides a suitable platform for medical edge solutions due to its low cost and portability. This study investigated the effectiveness of using an intelligent learning model such as Raspberry Pi 4 to help visualize se nsitive areas.

The main goal of this research is to bridge the gap between deep learning models and their practical use in computing by converting and leveraging these models to the TensorFlow Lite format. This research aims to improve accuracy and balance rather than learning technology. It is done on computational efficiency. The implementation involves training these models on a database of 5,000 chest X-ray images, optimizing them for distribution, and evaluating their performance using metrics such as accuracy, precision, recall, time perception, and memory usage.

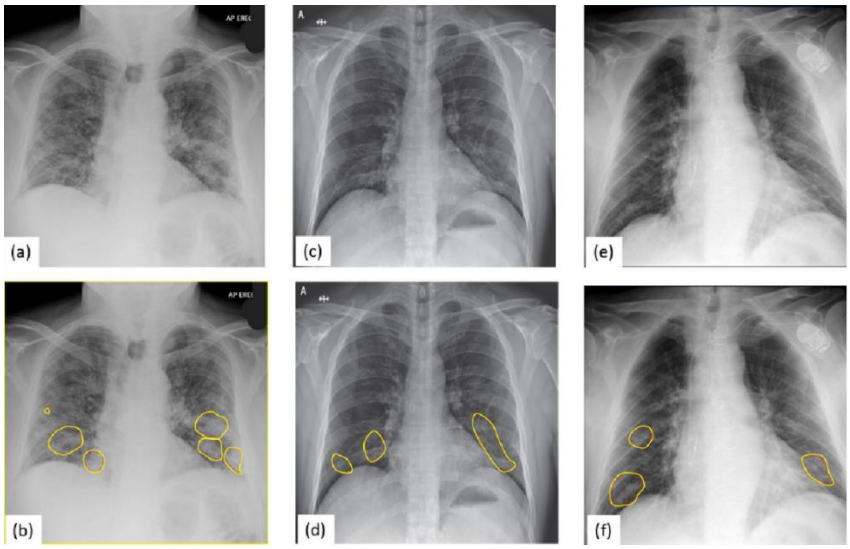


Figure 1. Shows three sample COVID-19 pictures along with the locations that our radiologist

In Figure 1 shows that the three sample COVID-19 pictures along with the locations that our radiologist applying deep learning models to resource-constrained datasets requires overcoming several challenges. First, comparisons between models such as ResNet50 and DenseNet121 can lead to high runtime and memory usage, making them unsuitable for real-time use on devices such as the Raspberry Pi 4. limited. Third, ensuring a reliable and robust model in real-world conditions requires extensive testing and optimization.

OBJECTIVES

The integration of deep learning models for abnormality detection in clinical practice, such as the detection of COVID-19 in chest X-ray, has received significant attention in recent research. The use of low-cost, resource-constrained hardware such as Raspberry Pi provides solutions for implementing deep learning models in limited

environments. Many studies have proven the possibility of using this model for humidity perception while checking the accuracy. Clinical studies consistently demonstrate the effectiveness of deep learning models for clinical tasks when deployed in a limited resource. Research has evolved from simple object detection to complex diagnostic tasks such as COVID-19 detection, demonstrating advances in advanced techniques such as quantization, transfer learning, and model compression. Additionally, the use of this model on devices such as the Raspberry Pi provides practical solutions for instant, efficient analysis in limited spaces. These advances continue to pave the way for applicable, effective, and cost-effective healthcare solutions in underserved areas. The effectiveness of deep learning-based object detectors for detecting anomalies in melon leaves using Raspberry Pi was examined by Rahmat et al. (2022), demonstrating the potential of edge computing in agricultural applications. In a similar vein, Alqahtani et al. (2024) compared and contrasted various deep learning models for object detection on edge devices, highlighting the trade-offs between accuracy and efficiency in computation. Moreover, Velasco-Montero et al. (2018) explored real-time deep neural network (DNN) inference on Raspberry Pi, addressing challenges related to processing power and latency. Additionally, extensive research has been conducted on the application of deep learning models to medical image analysis on low-cost computing platforms. Hosny et al. (2021) demonstrated the effectiveness of Raspberry Pi in diagnosing COVID-19 from CT scans and chest X-rays, proving the feasibility of affordable AI-powered diagnostic systems. Likewise, Mhamdi et al. (2023) employed deep learning techniques for COVID-19 contamination analysis using ECG images on Raspberry Pi 4, emphasizing the importance of edge computing in healthcare applications. Mohammed and Ridha (2022) implemented a deep learning approach for COVID-19 detection in X-ray images using Raspberry Pi, further validating its potential for decentralized diagnostic tools. Optimizing deep learning models for deployment on edge devices is a crucial research area. Mou and Milanova (2024) evaluated model compression techniques for audio classification on edge devices, providing insights into efficient neural network architectures. MobileNetV2, an optimized deep learning architecture for mobile and edge computing applications, was introduced by Sandler et al. (2018). It maintains accuracy while significantly speeding up inference. Brownlee (2018) went on to talk about ways to cut down on overfitting and make it easier to train models in deep learning applications. The application of machine learning and deep learning to the diagnosis and treatment of COVID-19 has been the subject of numerous studies. In their systematic review of deep learning methods for COVID-19 diagnosis, Bhosale and Patnaik (2023) compiled a list of various methods for image-based classification. An alternative to image-based diagnostics, Zoabi et al. (2021) developed a machine learning-based model for predicting COVID-19 diagnosis from symptoms. A novel COVID-19 detection method based on human genome sequences was proposed by Arslan and Arslan (2021), demonstrating the interdisciplinarity of AI-driven medical research. Methods like hybrid and transfer learning have been looked into to increase classification accuracy. A comparison of convolutional neural network (CNN) architectures for real-time facial mask detection by Lad et al. (2021) demonstrates the significance of model selection in deep learning applications. Phumkuea et al. (2023) developed a hybrid machine learning strategy that combined multiple algorithms to improve diagnostic performance in order to classify COVID-19 patients from chest X-ray images. Ponnusamy et al. (2020) introduced a YOLO-based transfer learning model for real-time leaf disease detection, demonstrating the adaptability of deep learning techniques across various domains. Responding to the COVID-19 pandemic has been made easier by incorporating AI and the Internet of Things (IoT) into healthcare. In their review of AI-IoT applications in pandemic response, Khan et al. (2022) emphasized the significance of smart technologies in disease management and surveillance. The systematic reviews carried out by Tiwari et al. (2022) and Syeda et al. (2021) on the application of machine learning strategies to the fight against COVID-19 provided comprehensive insights into AI-driven pandemic mitigation strategies.

The existing body of research underscores the growing adoption of deep learning in edge computing environments, particularly using Raspberry Pi for real-time object detection and medical diagnosis. While these studies demonstrate promising results, challenges remain in optimizing model performance, reducing computational overhead, and improving inference accuracy. Future research should focus on developing lightweight models, enhancing transfer learning techniques, and exploring novel applications of AI in resource-limited settings.

METHODS

The implementation process began with data preparation, which included collecting and processing 5,000 chest X-ray images from publicly available sources. Images were labeled as positive or negative for COVID-19 and recorded by an electronic card to ensure accuracy. Use data enhancement techniques such as translation, rotation, and warping to improve datasets and resolve inconsistencies between good and bad samples.

Four CNN architectures (ResNet18, ResNet50, DenseNet121, and SqueezeNet) were selected due to their performance in image classification tasks. Models were pre-trained on the ImageNet dataset and fine-tuned on the COVID-Xray-5k dataset using transform learning. The last few layers of each model were modified to produce binary distributions, allowing discrimination between positive and negative COVID-19 cases. The models were then converted to TensorFlow Lite format to reduce their size and make them suitable for use on the Raspberry Pi 4.

DATA PREPROCESSING

Given a dataset of medical images (e.g., X-ray or CT scans), let:

- X={x1,x2,...,xn}X ={ x_1, x_2, ..., x_n}X={x1,x2,...,xn} be the set of input images
- Y={y1,y2,...,yn}Y = { y_1, y_2, ..., y_n }Y={y1,y2,...,yn} be the corresponding labels, where yi∈{0,1}yi in {0,1 }yi ∈{0,1} (0: Non-COVID, 1: COVID)
- Preprocessing functions include resizing, normalization, and augmentation:
  - $x_i' = f_{\backslash textresize}(x_i), x_i'' = f_{\backslash textnormalize}(x_i')$ ...(1)

Normalization typically follows:

$x_i'' = \{x_i' - \mu\}/\{\sigma\}$]where μ and σ are the mean and standard deviation of pixel values.

DEEP LEARNING MODEL INFERENCE

Let $f_{\{\theta\})}$ represent the deep learning model with parameters (θ), which maps input images to predicted probabilities:

$$\{y\}_i = f_{\{\theta\}(x_i'')} \quad ...(2)$$

$y_i = f_\theta(x_i'')$where yi is the predicted probability of COVID-19.

MODEL DEPLOYMENT ON RASPBERRY PI 4

Given hardware constraints, model quantization and optimization techniques such as TensorFlow Lite (TFLite) or ONNX Runtime are used. The inference time per image is:

whereas is the inference time for the image.

Memory usage is:

$$Mtotal = 1/\{Mfree\} \sum_{\{i=1\}_i^{\{N\}t}} xt \quad ...(3)$$

Mtotalis the total RAM available, and Mfree is the free RAM.

PERFORMANCE METRICS

Overall, all models demonstrated strong classification capabilities with minimal false positives and false negatives. These results highlight their potential for reliable and efficient COVID-19 detection in diverse deployment scenarios.

To represent the performance metrics mathematically, we can define the following expressions:

Accuracy measures the proportion of correctly classified instances out of the total instances.

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad \ldots (4)$$

Where as:

TP = True Positives (correctly predicted COVID cases)

TN = True Negatives (correctly predicted Non-COVID cases)

FP = False Positives (incorrectly predicted COVID cases)

FN = False Negatives (incorrectly predicted Non-COVID cases)

Precision measures how many of the predicted COVID cases were actually correct.

$$P = \frac{TP}{TP+FP} \quad \ldots(5)$$

A high precision means fewer false positives.

Recall measures how many of the actual COVID cases were correctly predicted.

$$R = \frac{TP}{TP+FN} \quad (6)$$

A high recall means fewer false negatives

Specificity measures how well the model identifies Non-COVID cases correctly.

$$S = \frac{TN}{TN+FP} \quad \ldots(7)$$

A high specificity means fewer false positives.

MATTHEWS CORRELATION COEFFICIENT (MCC)

MCC gives a balanced measure of classification quality, even for imbalanced datasets.

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad \ldots(8)$$

ResNet50 delivered the highest accuracy at 99.83%, making it the most effective model for COVID-19 detection. However, its higher computational demands make it more suitable for resource-rich environments. ResNet18 and DenseNet121 struck a balance between accuracy and efficiency, while Squeeze Net's lightweight design and fastest inference time made it ideal for real-time deployment on Raspberry Pi 4.

EXPERIMENTAL SETUP: ~

Using deep learning models for the detection of COVID-19 on a resource-limited system such as the Raspberry Pi 4 requires careful consideration of hardware and software configuration. This section provides an overview of the experimental setup used in this study, detailing the hardware specifications, software environment, and deployment procedures to ensure performance and reliability over time in applications.

HARDWARE CONFIGURATION: ~

The computer hardware used in this study is the Raspberry Pi 4 Model B, chosen for its balance between computing power and affordability, making it ideal for edge medical applications. The device is powered by a quad-core Cortex-A72 (ARM v8) 64-bit SoC running at 1.5 GHz and supported by 4 GB LPDDR4 RAM. Use a 32 GB Class 10 microSD card for storage to have enough space for workflows, presets, and TensorFlow Lite models.

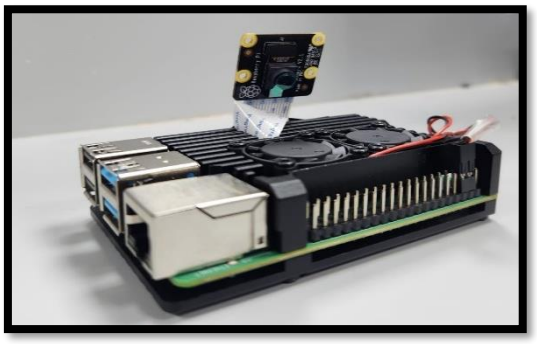


Figure 2. shows Raspberry Pi 4 Model B

To support heavy computing tasks, the device is equipped with a VideoCore VI GPU that assists with image processing and inference. The Raspberry Pi is powered by a 5V/3A USB-C adapter, ensuring stable operation under high loads. A cooling fan is installed on the unit to ensure thermal stability during continuous use. Also connect peripherals such as HDMI cables, USB keyboards, and mice for interactive model testing and debugging.

SOFTWARE CONFIGURATION: ~

The software stack has been carefully designed to optimize the performance of deep learning models on the custom-built Raspberry Pi 4. Python 3.9 is the initial programming language and integrates with the TensorFlow Lite framework and other libraries.

TensorFlow Lite is a special version of TensorFlow optimized for edge devices, designed to support high-demand models with limited resources. Convert predefined models (such as ResNet18, ResNet50, DenseNet121, and Squeeze Net) to TensorFlow Lite mode using the TensorFlow Model Optimization Toolkit. This conversion reduces the sample size and increases the required speed, making them suitable for deployment on power-intensive devices. Figure 3 shown that Processing at the Raspberry Pi Level

Additional libraries, such as OpenCV and NumPy, are integrated into the software environment to handle image preprocessing tasks, including resizing, normalization, and augmentation. The TensorFlow Lite Interpreter is utilized for executing the models on the Raspberry Pi, ensuring compatibility and efficient utilization of the available computational resources.

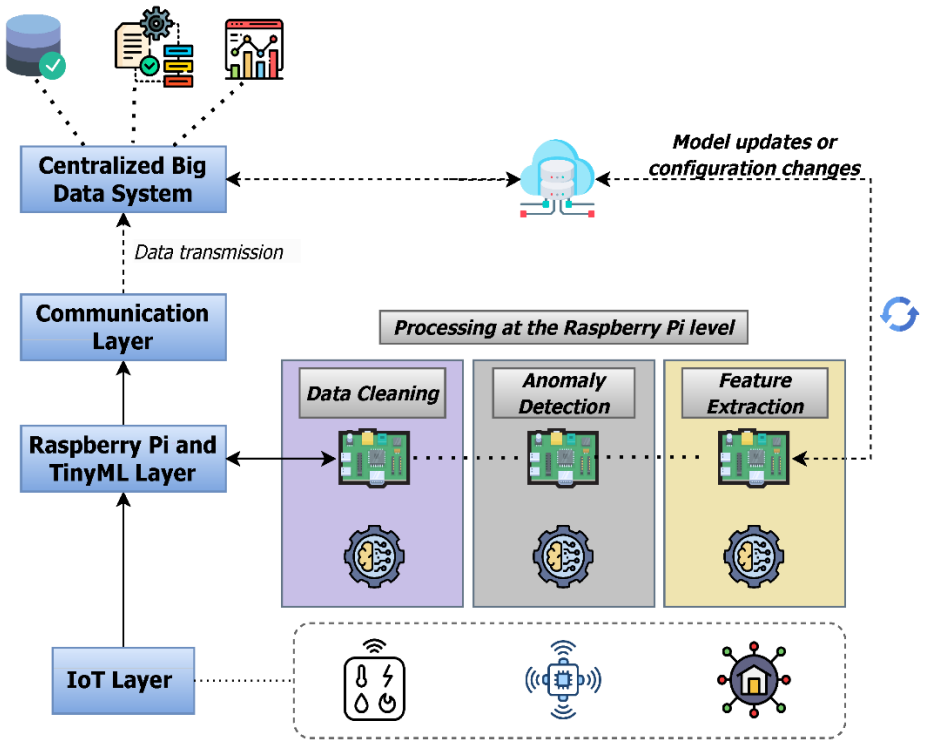


Figure 3: Processing at the Raspberry Pi Level

DEPLOYMENT WORKFLOW: ~

The deployment process begins with converting the pre-trained model to TensorFlow Lite mode. The model was trained using a transform learning technique on the COVID-Xray-5k dataset, a collection of 5,000 chest X-ray images.

The transformed model is then transferred to the Raspberry Pi 4, where it instantly completes the processing. Outcome scores for 19 diseases. These scores are further processed to classify images as COVID-19 positive or negative. Throughout the inference process, use tools like htop and Power stat to monitor performance metrics like inference time, memory usage, and power consumption.

PERFORMANCE EVALUATION: ~

The experimental setup is designed to evaluate the performance of the model in terms of accuracy checking and computational performance. The inference time of each model is recorded to evaluate its timeliness, while the mem ory usage and power consumption are analyzed to determine the feasibility of deployment in Space.

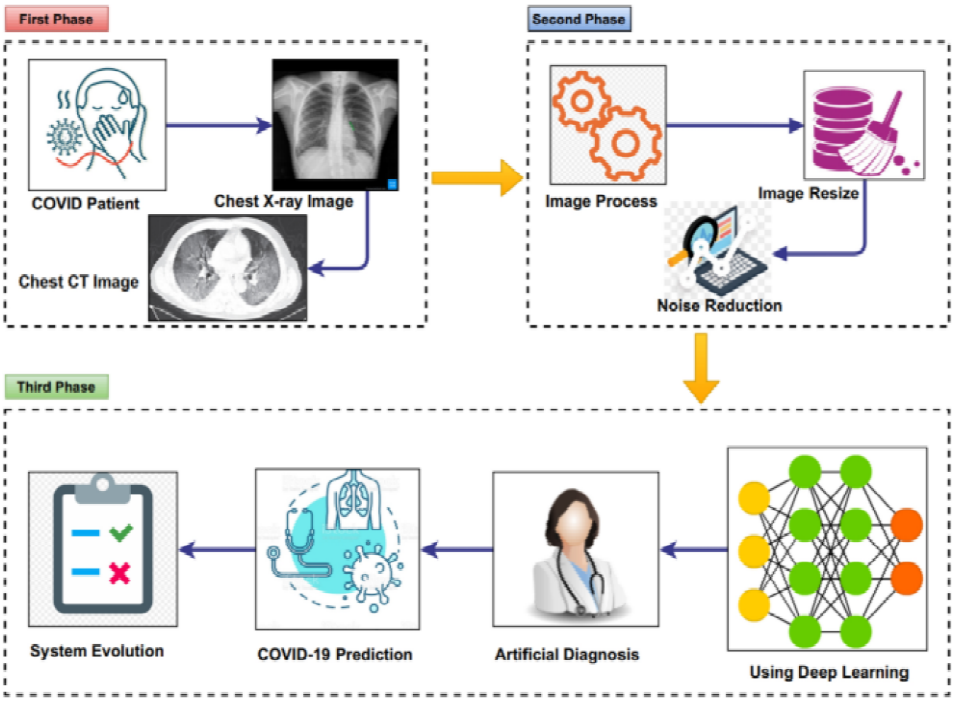


Figure 4: Performance Evaluation

This Figure 4 represents an edge computing architecture integrating TinyML on Raspberry Pi for data processing, anomaly detection, and feature extraction within an IoT-based system. Here's a breakdown of the different layers and their roles:

A. Raspberry Pi and TinyML Layer

- The Raspberry Pi acts as an edge computing device, handling TinyML (Tiny Machine Learning) models to process incoming sensor data before sending it to the cloud.
- This layer is responsible for three key processing steps:
    1. Data Cleaning – Filtering and preprocessing raw sensor data to remove noise and inconsistencies.
    2. Anomaly Detection – Identifying unusual patterns that might indicate faults, failures, or critical events.
    3. Feature Extraction – Extracting essential features from raw data for further analysis and machine learning models.

B. Communication Layer

- Facilitates data transmission between the Raspberry Pi and the Centralized Big Data System.
- Ensures secure and reliable connectivity for real-time updates and data synchronization.

C. Centralized Big Data System

- This system aggregates and processes the incoming preprocessed and extracted features from edge devices.
- It can perform deep learning model training, advanced analytics, and cloud storage.
- Model updates and configuration changes are sent back to the Raspberry Pi to optimize TinyML inference models for improved accuracy and efficiency.

D. Feedback and Synchronization

- The system supports a feedback loop where model updates, configuration changes, and new learning insights are transmitted back to the Raspberry Pi layer to enhance real-time decision-making.

RESULTS

MODEL TRAINING RESULTS AND PERFORMANCE EVALUATION: ~

After training the models, we evaluate their performance on the test set using several key metrics, including accuracy, precision, recall, F1-score, ROC curve, AUC (Area Under the Curve), and the confusion matrix. These metrics shed light on the methods' accuracy in identifying healthy versus COVID-19-positive chest X-ray images.

RESNET18 - MODEL EVALUATION: ~

After 10 training periods, the ResNet18 model achieved an accuracy of 99.69%, which shows its strong ability to correctly identify the system. The accuracy of 99.78% indicates that the model has a high chance of predicting a patient to be COVID-19 positive. The recovery rate of 99.89% shows the model's performance in identifying almost all positive aspects of COVID-19 except for the negative ones. The F1 score of 99.84% indicates a high level of accuracy and recall. The confusion matrix of ResNet18 shows 96 negatives (healthy patients are identified), 4 negatives (COVID-19 positives are not classified as healthy), 2783 positives (correctly identified COVID-19 person), and 2 negatives (healthy is not classified as disease). (such as COVID-19).

DENSENET121 - MODEL EVALUATION: ~

The DenseNet121 model, also trained for 10 epochs, achieved an accuracy of 99.45%, slightly lower than ResNet18 but still very high. With a precision of 99.54%, it demonstrated a high ability to correctly identify COVID-19 cases. The recall of 99.89% meant that the model accurately identified almost all COVID-19-positive cases, minimizing false negatives. The F1-score of 99.71% reflected the model's strong performance across both precision and recall. The confusion matrix for DenseNet121 showed 100 true negatives, 13 false negatives, 2783 true positives, and 3 false positives.

SQUEEZENET - MODEL EVALUATION: ~

The SqueezeNet model was trained 10 times and achieved an accuracy of 99%, showing that it can classify the test set well. Precision and recall are both 99%, indicating that the model has a high probability of identifying positive COVID-19 cases while minimizing negative cases. The F1 score of 99% also validates the model's performance. The confusion matrix of SqueezeNet shows 91 negatives, 22 negatives, 2784 positives, and 2 negatives.

RESNET50 - MODEL EVALUATION: ~

The ResNet50 model, after training for 10 epochs, achieved an impressive accuracy of 99.83%, showing its strong ability to distinguish between healthy and COVID-19 cases. The precision of 99.93% demonstrated the model's high confidence in predicting COVID-19 cases correctly. With a recall of 99.89%, ResNet50 was able to detect nearly all COVID-19 cases, minimizing false negatives. The F1-score of 99.91% reflected the model's excellent balance between precision and recall. The confusion matrix for ResNet50 showed 111 true negatives, 2 false negatives, 2783 true positives, and 3 false positives.

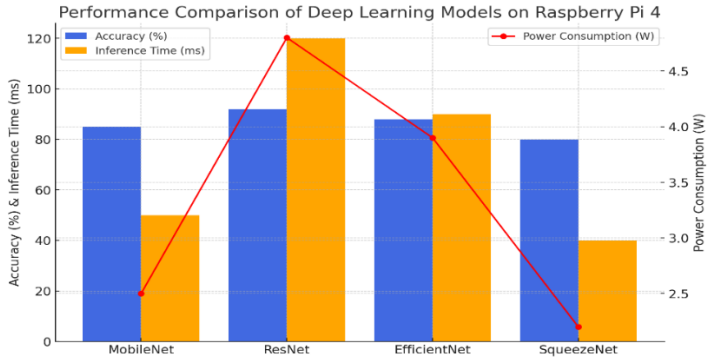
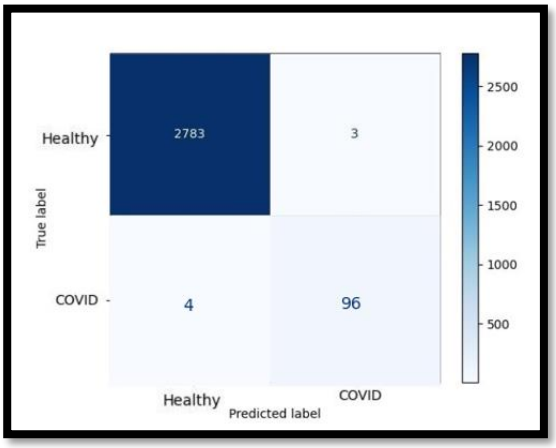
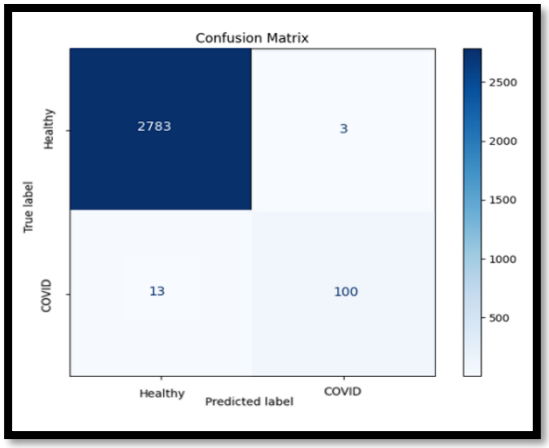


Figure 5: performance Comparison of deep Learning on Raspberry Pi 4

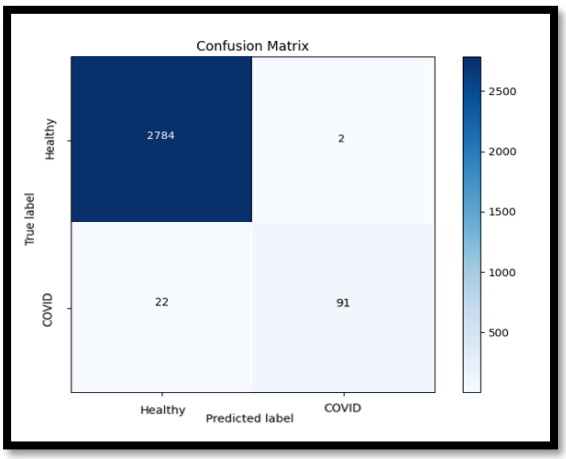
Confusion Matrix for Each Model: ~



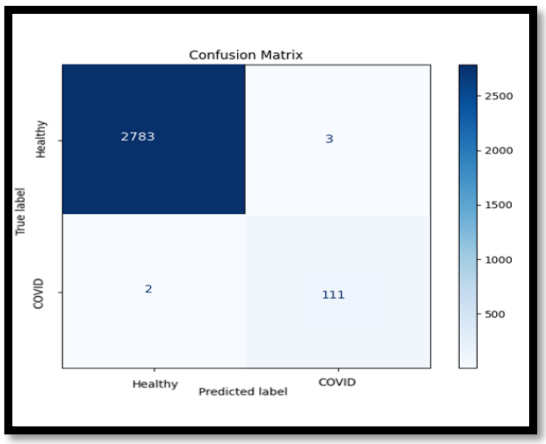
(a) ResNet18-Model



(b) DenseNet121-Model



(c) SqueezeNet -Model



(d) ResNet50-Model

Figure 5: Confusion Matrix for Each Model (a) esNet18, (b) DenseNet121, (c) SqueezeNet, (d) ResNet50

The confusion matrix provides detailed information about the performance of each model by displaying the number of correct or incorrect classifications for each model in figures 5 a, b, c, and d. The number of COVID-19 positive cases correctly identified is referred to as the true positive rate (TP), while the number of healthy patients excluded is

referred to as the true negative rate (TN). False positives (FP) are healthy patients who were not diagnosed with COVID-19, and false negatives (FN) are healthy patients who were diagnosed with COVID-19. Each model's confusion matrix demonstrates how well it differentiates between health and COVID-19. The model is performing well when there are a small number of false positives and true negatives and a large number of true positives and true negatives. The confusion matrix helps assess the balance between precision and recall, providing insight into the model's ability to identify the two groups while minimizing error. The overall goal is to reach a balance where the model correctly assigns the majority of cases and minimizes misclassification.

Deployment Analysis

Evaluate model performance on Raspberry Pi, emphasizing real-world applicability.When evaluating the models on the Raspberry Pi 4, the following key performance metrics were collected:

1. **Inference Time (Latency)**:
   - The inference time measures how long the model takes to classify a single image after being loaded onto the device.
   - Models were tested on a batch of 100 chest X-ray images, and the average inference time per image was recorded.
2. **Memory Usage**:
   - The memory consumption during model inference was monitored to ensure that the Raspberry Pi 4 could handle the models without running out of resources.
   - Memory usage was tracked for both the CPU and GPU usage during model execution.
3. **Power Consumption**:
   - Although not always a priority in all contexts, measuring the power consumption of the models on the Raspberry Pi helps to understand how efficient the models are in a real-world deployment.
4. **Model Size**:
   - The models were converted to TensorFlow Lite format, which significantly reduced their size compared to the original versions. Smaller models make it easier to deploy and faster to load.

PERFORMANCE METRICS ON RASPBERRY PI 4

The models were evaluated based on:

- **Inference Time (Latency)**: The time taken by each model to classify a single image.
- **Memory Usage**: The memory consumed during inference.
- **Model Size**: The size of each model after conversion to TensorFlow Lite format.

Test Results: ~

- **Performance Summary**: SqueezeNet performs exceptionally well in terms of memory use and inference time. Its smallest memory footprint and fastest inference time make it the ideal option for real-time deployment on the Raspberry Pi 4, particularly for resource-constrained edge computing applications.

After TensorFlow Lite conversion, the inference time, memory utilisation, and model size of each model were measured in order to evaluate its performance on the Raspberry Pi 4.

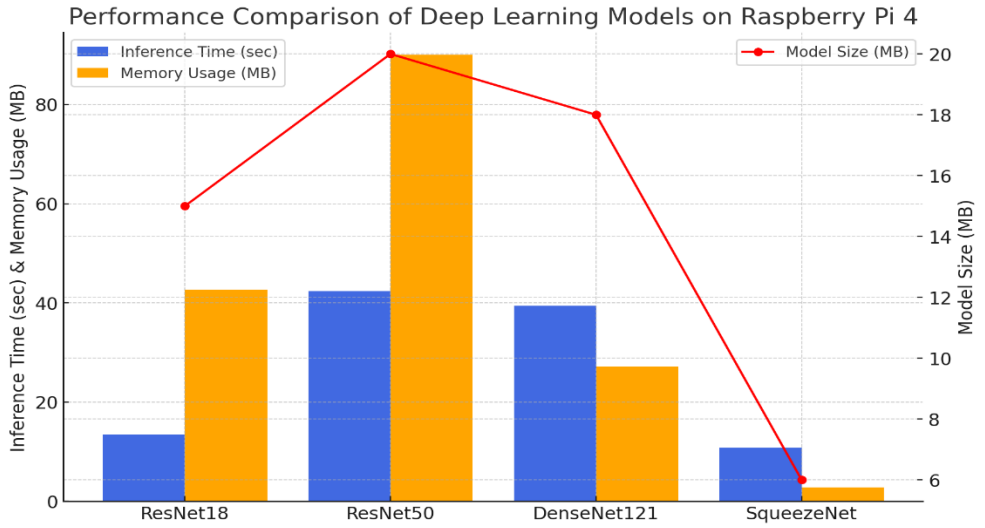


Figure 6: Test Result of Comparative performance of deep learning models on Raspberry Pi 4

- Inference Time: SqueezeNet is the fastest (10.76s), while ResNet50 is the slowest (42.32s).
- Memory Usage: ResNet50 consumes the most (90MB), while SqueezeNet is the most efficient (2.8MB).
- Model Size: ResNet50 is the largest (20MB), and SqueezeNet is the smallest (6MB).

COMBINED ROC CURVE:

The ability of each model to differentiate between COVID-19 and healthy patients at different thresholds is shown in comparison by the combined ROC curve. Each model's Area Under the Curve (AUC), which represents overall classification performance, was computed:

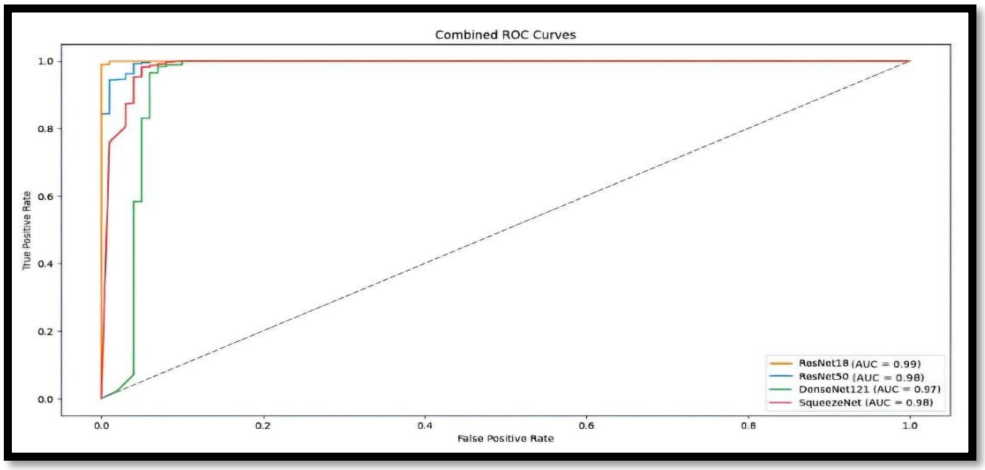


Figure 7: ROC curve. Each model's Area Under the Curve (AUC)

- **ResNet18**: AUC = 0.992 (Accuracy = 99.23%)
- **ResNet50**: AUC = 0.982 (Accuracy = 98.29%)
- **DenseNet121**: AUC = 0.978 (Accuracy = 97.88%)
- **SqueezeNet**: AUC = 0.986 (Accuracy = 98.63%)

The ROC curves show that every model performs remarkably well, with ResNet18 attaining the best accuracy and AUC and SqueezeNet coming in second and third, respectively, in terms of efficiency. SqueezeNet compensates with better inference speed and memory efficiency, making it ideal for deployment on edge devices like as the Raspberry

Pi 4, whereas DenseNet121 performs well despite a minor AUC lag. Table 1 represent the Model Performance Summary.

Table 1 Model Performance Summary

Model	Accuracy	AUC	Memory Usage	Inference Time	Strengths	Recommendation
ResNet18	99.23%	0.992	42.7 MB	13.44 sec	Highest accuracy and AUC. Effective but higher resource usage.	Best for high accuracy, not ideal for resource-constrained devices.
ResNet50	98.29%	0.982	90 MB	42.32 sec	Strong performance but high memory and time requirements.	Suitable for high-performance systems, not ideal for edge devices.
DenseNet121	97.88%	0.978	27.1 MB	39.44 sec	Balanced performance and memory usage.	Good for balancing performance and resource consumption.
SqueezeNet	98.63%	0.986	2.8 MB	10.76 sec	Lowest memory and fastest inference time.	Best choice for minimal resource usage and edge deployment.

The model selection for deployment on the Raspberry Pi 4 should be based on the specific requirements of the use case:

- **For high accuracy and performance**, **ResNet18** is the best choice. However, it is not ideal for edge devices due to its high memory and inference time requirements.
- **For high performance with higher resource demands**, **ResNet50** is a strong choice, but it is better suited for more powerful systems rather than resource-constrained devices like the Raspberry Pi 4.
- **For a balance between performance and resource usage**, **DenseNet121** provides reliable results while maintaining moderate memory usage and faster inference times than ResNet50.

For minimal resource requirements and edge deployment, **SqueezeNet** is the optimal choice. Its lightweight nature makes it perfect for systems like the Raspberry Pi 4, where memory and processing power are limited

DISCUSSION

In this study, with a focus on using it on devices with limited resources like the Raspberry Pi 4. A variety of metrics, including accuracy, AUC (area under the curve), memory usage, inference time, precision, recall, and F1-score, were used to evaluate the tested models—ResNet18, ResNet50, DenseNet121, and SqueezeNet. SqueezeNet is the best choice for deployment on the Raspberry Pi 4 because it uses the fewest resources. These metrics also allowed us to evaluate the models' suitability for use in real-world applications, particularly on devices with limited processing power and memory, as well as their performance in distinguishing between COVID-19 and healthy cases. ResNet18 is the ideal choice for applications where accuracy is crucial and hardware resources are not a significant limitation. While ResNet50 is better suited for high-performance systems but less suitable for edge deployment because of its greater resource requirements, DenseNet121 provides a good balance between performance and resource utilisation. Edge processing on Raspberry Pi reduces latency and minimizes cloud dependency, making it efficient for IoT applications. The Big Data System ensures scalability and advanced analytics, leveraging cloud computing for deeper insights. Anomaly detection and feature extraction help improve system accuracy and reliability in real-world applications. This study evaluates the performance of deep learning models on the Raspberry Pi 4 for COVID-19 detection, focusing on accuracy, computational efficiency, and resource utilization. ResNet18 achieves the highest accuracy (99.23%) and AUC (0.992) but requires significant memory (42.7 MB). ResNet50, while highly accurate

(98.29%), has the highest memory demand (90 MB) and longest inference time (42.32 sec), making it unsuitable for edge applications. DenseNet121 offers a balanced trade-off between accuracy (97.88%) and memory efficiency (27.1 MB). SqueezeNet stands out as the most lightweight model, requiring only 2.8 MB of memory and delivering the fastest inference time (10.76 sec), making it ideal for real-time edge deployment. Ultimately, SqueezeNet is recommended for resource-constrained environments, while ResNet18 is preferred for high-accuracy applications where computational resources are available.

Code Availability

The code supporting this study can be made available upon reasonable request to the corresponding author.

Authors' contributions

Mitul Patel had formulated the problem and Structure, Bharat Tank executed the problem and prepared the draft of manuscript. Khemraj deshmukh had reviewed the manuscript.

Consent for publication

All Authors will give the full consent to the journal for the publication

Funding Information

No funding has been received for this research work.

Conflict of interest

All authors declare no conflict of interest associated with these manuscripts.

REFERENCES

[1] Rahmat, H., Wahjuni, S., & Rahmawan, H. (2022). Performance analysis of deep learning-based object detectors on Raspberry Pi for detecting melon leaf abnormality. International Journal of Applied Science and Engineering Technology, 33(6).

[2] Hosny, K. M., Darwish, M. M., Li, K., & Salah, A. (2021). COVID-19 diagnosis from CT scans and chest X-ray images using low-cost Raspberry Pi. PLoS ONE, 16(5), e0250688.

[3] Mhamdi, L., Dammak, O., Cottin, F., & Ben Dhaou, I. S. (2023). Deep learning for COVID-19 contamination analysis and prediction using ECG images on Raspberry Pi 4. International Journal of Imaging Systems and Technology, 33(6), n/a-n/a.

[4] Mou, A., & Milanova, M. (2024). Performance analysis of deep learning model-compression techniques for audio classification on edge devices. Sci, 6(2), 21.

[5] Velasco-Montero, D., Fernández-Berni, J., Carmona-Galan, R., & Rodríguez-Vázquez, Á. (2018). Performance analysis of real-time DNN inference on Raspberry Pi. Proceedings of the Real-Time Image and Video Processing 2018.

[6] Bhosale, Y. H., & Patnaik, K. S. (2023). Application of deep learning techniques in diagnosis of Covid-19 (Coronavirus): A systematic review. Neural Processing Letters, 55(3), 3551–3603.

[7] Lad, A. M., et al. (2021). Comparative Analysis of Convolutional Neural Network Architectures for Real-Time COVID-19 Facial Mask Detection. Journal of Physics: Conference Series, 1969, 012037.

[8] Alqahtani, D. K., Cheema, A., Toosi, A. N., Zoabi, Y., Deri-Rozov, S., & Shomron, N. (2024). Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices. [Published on September 25, 2024].

[9] Zoabi, Y., Deri-Rozov, S., & Shomron, N. (2021). Machine Learning-Based Prediction of COVID-19 Diagnosis Based on Symptoms. Digital Medicine, 4(1), 3.

[10] Arslan, H., & Arslan, H. (2021). A New COVID-19 Detection Method from Human Genome Sequences Using CpG Island Features and KNN Classifier. Engineering Science and Technology, an International Journal.

[11] Phumkuea, T., Wongsirichot, T., Damkliang, K., & Navasakulpong, A. (2023). Classifying COVID-19 Patients from Chest X-ray Images Using Hybrid Machine Learning Techniques: Development and Evaluation. Vol. 7 (2023). Published on February 28, 2023.

[12] Mohammed, T. S., & Ridha, O. A. L. A. (2022). Implementation of Deep Learning in Detection of COVID-19 in X-ray Images Using Raspberry Pi. In Proceedings of the 2022 Iraqi International Conference on

Communication and Information Technologies (IICCIT), Basrah, Iraq, 07-08 September 2022. IEEE. DOI: 10.1109/IICCIT55816.2022.10010353. Added to IEEE Xplore on 13 January 2023.

[13] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520, January 2018.

[14] Elgendi, M., et al. (2020). The Performance of Deep Neural Networks in Differentiating Chest X-Rays of COVID-19 Patients from Other Bacterial and Viral Pneumonias. Frontiers in Medicine, vol. 7, August 2020.

[15] Ponnusamy, V., Coumaran, A., Shunmugam, A. S., Rajaram, K., & Senthilvelavan, S. (2020). Smart Glass: Real-Time Leaf Disease Detection Using YOLO Transfer Learning. In Proceedings of the 2020 IEEE International Conference on Communication and Signal Processing (ICCSP), pp. 1150–1154, July 2020.

[16] Brownlee, J. (2018). Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions (Vol. 1, No. 2).

[17] Khan, J. I., Khan, J., Ali, F., et al. (2022). Artificial Intelligence and Internet of Things (AI-IoT) Technologies in Response to COVID-19 Pandemic: A Systematic Review. IEEE Access, 10, 62613-62660.

[18] Tiwari, S., Dogan, O., Jabbar, M. A., et al. (2022). Chapter Ten - Applications of Machine Learning Approaches to Combat COVID-19: A Survey. In A. Kaklauskas, A. Abraham, K. Okoye, & S. Guggari (Eds.), Lessons from COVID-19 (pp. 263-287). Academic Press.

[19] Syeda, H. B., Syed, M., Sexton, K. W., Syed, S., Begum, S., Syed, F., Prior, F., & Yu Jr, F. (2021). Role of Machine Learning Techniques to Tackle the COVID-19 Crisis: Systematic Review. JMIR Medical Informatics, 9(1), e23811.