Neural Networks Based Smart E-Health Application for the Prediction of Tuberculosis Using Serverless Computing

Subramaniam Subramanian Murugesan, Sasidharan Velu, Muhammed Golec, Huaming Wu, *Senior Member, IEEE*, and Sukhpal Singh Gill

***Abstract*—The convergence of the Internet of Things (IoT) with e-health records is creating a new era of advancements in the diagnosis and treatment of disease, which is reshaping the modern landscape of healthcare. In this paper, we propose a neural networks-based smart e-health application for the prediction of Tuberculosis (TB) using serverless computing. The performance of various Convolution Neural Network (CNN) architectures using transfer learning is evaluated to prove that this technique holds promise for enhancing the capabilities of IoT and e-health systems in the future for predicting the manifestation of TB in the lungs. The work involves training, validating, and comparing Densenet-201, VGG-19, and Mobilenet-V3-Small architectures based on performance metrics such as test binary accuracy, test loss, intersection over union, precision, recall, and F1 score. The findings hint at the potential of integrating these advanced Machine Learning (ML) models within IoT and e-health frameworks, thereby paving the way for more comprehensive and data-driven approaches to enable smart healthcare. The best-performing model, VGG-19, is selected for different deployment strategies using server and serless-based environments. We used JMeter to measure the performance of the deployed model, including the average response rate, throughput, and error rate. This study provides valuable insights into the selection and deployment of ML models in healthcare, highlighting the advantages and challenges of different deployment options. Furthermore, it also allows future studies to integrate such models into IoT and e-health systems, which could enhance healthcare outcomes through more informed and timely treatments.**

***Index Terms*—e-health, healthcare, IoT, machine learning, predictive models, serverless computing, tuberculosis.**

Manuscript received 18 September 2023; revised 29 January 2024; accepted 16 February 2024. Date of publication 20 February 2024; date of current version 6 September 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62071327 and in part by Tianjin Science and Technology Planning Project under Grant 22ZYYYJC00020. *(Corresponding author: Huaming Wu.)*

Subramaniam Subramanian Murugesan, Sasidharan Velu, Muhammed Golec, and Sukhpal Singh Gill are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS London, U.K. (e-mail: s.subramanianmurugesan@se22.qmul.ac.uk; s.velu@se22.qmul.ac.uk; m.golec@qmul.ac.uk; s.s.gill@qmul.ac.uk).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Digital Object Identifier 10.1109/JBHI.2024.3367736

I. Introduction

THE rapid advancements in Machine Learning (ML) and Artificial Intelligence (AI) technologies have significantly impacted the field of medical diagnosis, particularly in the realm of pulmonary diseases such as Tuberculosis (TB) [1]. Moreover, the integration of these cutting-edge technologies has paved the way for revolutionary developments in the Internet of Things (IoT) and e-health solutions [2], [3], [4], [5]. These innovations allow for real-time monitoring of patients' health conditions and the seamless transmission of medical data to healthcare providers [6], enabling more timely and accurate diagnoses and treatments [7]. TB is a contagious ailment primarily impacting the lungs, resulting from specific bacteria. It is transmitted when individuals with the infection cough, sneeze, or release saliva into the air. TB can be both prevented and treated. In 2021, TB resulted in 1.6 million fatalities, inclusive of 187,000 individuals with HIV. Globally, TB ranked as the 13th primary cause of death, and it was the second-highest infectious cause of death after COVID-19, surpassing HIV/AIDS. Around 10.6 million individuals worldwide contracted TB that year, comprising 6 million men, 3.4 million women, and 1.2 million children. Although TB affects all nations and ages, it's notable that it can be both treated and averted. MultiDrug-Resistant TB (MDR-TB) continued to be a significant public health concern in 2021, with only a third of those afflicted receiving treatment. From 2000 to 2021, approximately 74 million lives were rescued due to TB treatments and diagnoses. To meet the global goals established at the 2018 UN summit on TB, an annual funding of US$13 billion is required to support TB care, prevention, diagnosis, and treatment. One of the health objectives of the UN's Sustainable Development Goals (SDGs) is to eradicate the TB epidemic by 2030[1]. The diagnosis of TB typically requires careful interpretation of chest radiographs, a process that can be both time-consuming and subject to human error. The automation of this process using ML technologies, integrated with IoT and E-Health systems in the future, promises to increase both the speed and accuracy of TB diagnosis[2].

Image recognition technology has evolved significantly, finding applications across various fields and technologies. Its fundamental goal is to enable computers to process and interpret visual data similarly to human vision, but with greater speed and accuracy. Image recognition technology has evolved significantly, finding applications across various fields and

[1]https://www.who.int/news-room/fact-sheets/detail/tuberculosis
[2]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9041161/

technologies. Its fundamental goal is to enable computers to process and interpret visual data similarly to human vision, but with greater speed and accuracy. Researchers have shown that a pure transformer architecture can be useful for image recognition tasks and can get great results with less computing power than convolutional networks [8]. Further, the implementation of Oscillatory Neural Networks (ONNs) using Field-Programmable Gate Array (FPGA) for digit recognition from a camera stream highlights the adaptability of image recognition in real-time applications [9]. Moreover, Genetic Algorithm Augmented Convolutional Neural Network (CNN) has been utilised to improve performance in training time and accuracy [10]. Another study discussed making an Advanced Driver Assistance System (ADAS) with a heterogeneous multicore System on Chip (SoC) that has dedicated accelerators for different image recognition tasks [11]. Furthermore, the Sequence-to-Sequence Domain Adaptation Network (SSDAN) for robust text image recognition [12] and image recognition for the detection of Distribuited Denial of Service (DDoS) malware in IoT environments [13] and applied transfer learning techniques for image recognition in categorising nanoscience images obtained by scanning electron microscope [14]. Finally, a comprehensive review of deep learning methods in plant phenotypic image recognition is conducted to understand the literature in detail [15].

This paper presents a comparative study of three different CNN architectures, namely, DenseNet201, VGG19, and MobileNet V3 Small. The research leverages transfer learning, a technique that utilizes pre-trained models to bypass the need for large amounts of data and computational resources [16]. Performance evaluation of these models is conducted using two publicly available chest radiograph datasets released by the U.S. National Library of Medicine [17]. These datasets, comprising normal and abnormal chest X-rays, are sourced from Montgomery County's TB screening program and the Shenzhen No. 3 People's Hospital in China. Post model selection, the research investigates the comparison of two deployment strategies: a traditional server-based deployment on an AWS EC2 instance, and a serverless approach using AWS Fargate and AWS Elastic Kubernetes Service (EKS) for containerized deployment. When we refer to "serverless" in the context of AWS Fargate and AWS EKS, it doesn't mean that servers aren't used at all. Instead, "serverless" in this context means that the developers and operators don't need to provision, manage, or maintain the underlying servers [18]. The cloud provider (in this case, AWS) abstracts away the infrastructure management. Application developers can use serverless computing without handling backend infrastructure. This paradigm emphasises front-end and business development. The cloud provider is in charge of managing all backend processes, including infrastructure scaling and maintenance. This strategy streamlines development and delivers a cost-effective charging mechanism based on real usage, avoiding idle resource expenses.

A. Motivation and Our Contributions

TB is an infectious disease that carries the risk of death, and even after recovery, there is a risk of re-infection [19]. Chest radiography, which is the most commonly used diagnostic tool to diagnose TB, has an important value in identifying the disease. This has led to increased interest in the field of medical imaging research, particularly in the automated detection of chest diseases using lung radiographs [20]. Accurate diagnosis of active TB is crucial for an effective TB control initiative. Patients who remain undiagnosed with TB remain infectious and at risk of death; Patients who do not have TB but are misdiagnosed are unnecessarily exposed to potentially harmful drugs, wasting valuable public health resources. Additionally, only a small fraction of MD-R TB cases are confirmed by laboratories; This highlights the need for adequate diagnostic capacity for all forms of drug-resistant tuberculosis to advance global TB care. Therefore, TB control strategies should prioritize early diagnosis and appropriate treatment for all types of TB [21].

This work contributes to ongoing efforts to automate TB diagnosis by introducing a new architecture based on serverless computing and ML. The novelty of this work lies in the comprehensive approach to TB diagnosis by combining advanced CNN architectures with innovative delivery strategies. This dual focus not only increases the accuracy of disease detection but also streamlines the implementation process, making it more adaptable and efficient to real-world scenarios. Furthermore, our research is pioneering in integrating serverless computing paradigms into healthcare, potentially revolutionizing the speed and efficiency of disease diagnosis and management. To achieve this, we have selected Densenet-201, VGG-19, and Mobilenet-V3-Small architectures, because other well-known research works [22], [23] have shown that these models are most suitable for understanding the patterns, weights, and biases in a greyscale image such as Chest X-Rays. Results from this research could potentially lead to the development of robust, scalable, and accurate computer-aided diagnostic systems for TB, facilitating more efficient patient treatment. The first part of the innovative approach involves choosing the right ML model. Following model selection, two different deployment strategies are explored: AWS EC2 (traditional server-based deployment) and (serverless-based deployment) using AWS Fargate and AWS EKS. The paper concludes by showing the performance superiority of serverless computing over traditional servers by comparing two different deployment strategies. This study provides new insights into the selection and deployment of ML models for disease detection, paving the way for more efficient diagnostic systems. The main contributions of this paper are:

- Emphasizes the importance of chest radiography in disease detection.
- Enables real-time monitoring and early detection of TB with a proactive approach by integrating with IoT.
- Facilitates TB diagnosis using advanced CNN architectures.
- Explores and compares distribution strategies, highlighting latency considerations.

The rest of the paper is structured as follows. Section II presents the studies on ML models and their practices to predict Pneumonia-based diseases and their deployment in the real world. Section III describes the proposed methodology. Section IV presents the performance evaluations and results. Section V concludes and highlights future directions.

II. Related Work

In this section, we delve deeply into various research initiatives where ML, particularly CNN architectures, has been

TABLE I
COMPARISON OF PROPOSED WORK WITH EXISTING STUDIES

Study	Disease	AI	Deployment	Scalability
[16] [22] [23] [27]	COVID-19	✓	×	×
[24] [25]	Pneumonia	✓	×	×
[26]	Tuberculosis	✓	×	×
Our Work (this paper)	Tuberculosis	✓	✓	✓

utilized for interpreting lung X-ray images. The primary objective of these studies is to diagnose lung-related diseases, highlighting the efficacy of deep-learning models. Our focus is predominantly on TB diagnosis, aiming to not only evaluate the diagnostic capabilities of AI but also explore the operational facets of deployment and scalability. This approach intends to bridge the gap between academic research and real-world clinical applications. **COVID-19 Detection**: Showkat et al. [16] utilized the ResNet deep learning model to classify pneumonia from chest X-rays, achieving notable success in detecting COVID-19-related pneumonia. Shelke et al. [22] employed AI to analyze chest X-rays, achieving a remarkable 98.9% accuracy rate in distinguishing COVID-19 from normal pneumonia using the DenseNet-161 model. Tangudu et al. [23] introduced an optimized model with a 99% accuracy rate across two datasets for COVID-19 detection via chest radiographs, leveraging the MobileNet architecture.

Pneumonia Severity and Classification: Dey et al. [24] revealed that the VGG19 deep learning system, combined with an Ensemble Feature Scheme and Random-Forest classifier, achieves a 97.94% accuracy in diagnosing pneumonia from chest X-rays. Saleh et al. [25] utilized eight pre-trained models to discern pneumonia severity, with the MobileNet model achieving up to 94.23% accuracy.

Tuberculosis Detection: The study in [26] employed the ConvNet model to detect TB from chest X-rays, achieving an 87% accuracy rate, although pre-trained models like Xception achieved slightly higher precision. Our study distinguishes itself by emphasizing deployment strategies, an aspect often overlooked in related research. We use Amazon Fargate with EKS to deploy our model serverlessly, making it practical and applicable. Table I compares our proposed work with existing studies. Most similar research focuses on model performance and accuracy but overlooks deployment and scaling solutions. Our work addresses this gap by focusing on real-world deployment strategies.

III. METHODOLOGY

This section explains comprehensive methodology implemented in our study, including the datasets, CNN models, transfer learning strategies applied to these models, system architecture, and deployment strategies.

A. Dataset

Our study employed two principal chest X-ray datasets: the Montgomery County set (MC) and the Shenzhen set, both of which were retrieved from the popular data science platform, Kaggle[3] [17]. The dataset used for this research was sourced from the National Library of Medicine, National Institutes of Health, Bethesda, MD, USA and Shenzhen No.3 People's Hospital, Guangdong Medical College, Shenzhen, China. **The Montgomery County set**, assembled in cooperation with the Department of Health and Human Services, Montgomery County, USA, comprises 138 frontal chest X-rays. Out of these, 80 are normal cases and 58 are cases showing indications of TB. All images were captured using an Eureka stationary X-ray machine and are provided in PNG format, with DICOM format also available upon request. These details, as well as subsequent analyses of the data, were published in [17]. **The Shenzhen dataset**, collected in collaboration with the Shenzhen No.3 People's Hospital, China, consists of 662 frontal chest X-rays, with 326 normal and 336 exhibiting signs of TB. These images were captured as part of routine hospital procedures and are provided in PNG format. This dataset and its insights were also discussed in [17]. Both datasets not only have a clear naming convention for ease of interpretation, with identifiers signifying whether an X-ray is normal or abnormal, but also come with a clinical reading that outlines crucial information such as the patient's age, gender, and any detected lung abnormalities [17]. In addition to [17], these datasets have also been used in research investigations [28], [29]. Considering the diversity and variability in medical imaging data, incorporating chest X-ray images from various hospitals could significantly enhance our model's applicability. Federated learning, where model training is decentralized and data remains at its original location, emerges as a viable solution. This method not only addresses privacy concerns but also allows for a richer, more diverse dataset without the need to transfer sensitive medical data [30], [31].

B. Model Selection

In this research, we employed a transfer learning approach that leverages pre-trained models to efficiently detect TB in a resource-efficient manner. The approach encompasses the following key aspects: Adaptation of Pre-Trained Models: We initiated our methodology by adapting pre-trained models for TB detection. This involved preparing datasets for both training and testing, followed by tailoring these models to specifically address the task of TB detection. *1) Feature Extraction and Fine-Tuning:* A critical phase of our approach was the feature extraction and subsequent fine-tuning. Initially, the model was trained and evaluated using the prepared datasets. This was followed by fine-tuning, where we unfroze layers in the base model and retrained it with a reduced learning rate. This fine-tuning process was crucial in enhancing the model's performance by achieving a deeper understanding of the dataset-specific features. *2) Optimization Strategies:* We fine-tuned our model for an additional five epochs with a reduced learning rate of 0.001. The decision to unfreeze the last 30% of the layers in each model was based on our findings and aligned with established guidelines [16], proving to be effective in optimizing the model's performance for TB detection. *3) Utilization of Specific Models:* In our research, we specifically employed models like DenseNet201 and VGG19, known for their proficiency in image recognition tasks, and adapted them for our TB classification task. This allowed us to leverage the extensive knowledge these models had acquired from training on large-scale image datasets, making them particularly suited for our research needs. This strategy enables us to leverage the knowledge these models have already

[3]https://www.kaggle.com/datasets/nikhilpandey360/chest-xray-masks-and-labels

acquired through training on a large-scale image dataset and tailor it to our TB classification task. Further elaboration on the individual models and the fine-tuning process will be presented in the following sections. The transfer learning approach we adopted was not only instrumental in enhancing the model's accuracy but also in conserving computational resources, making it a viable solution for integrating with IoT and e-health systems for future healthcare delivery.

1) Densenet 201: In our study, we draw relevance from the research detailed in [27] concerning the utilization of the DenseNet201 model for TB classification. This work outlines a DenseNet201-based deep transfer learning (DTL) model specifically for the diagnosis of COVID-19 patients, which has been instrumental in guiding our choice of models. Their methodology involves leveraging DenseNet201 for feature extraction, utilizing its pre-trained weights on the ImageNet dataset, and incorporating a convolutional neural structure. The demonstrated effectiveness of DenseNet201 in differentiating COVID-19 cases from chest CT scan images in their study provides compelling evidence for its prospective relevance in analogous situations, including our Tuberculosis case classification. The choice to include DenseNet-201 in our study was driven by its architectural strengths, notably, its dense connectivity pattern, which alleviates the vanishing-gradient problem, strengthens feature propagation, and encourages feature reuse. This makes DenseNet-201 highly effective for medical image analysis, particularly in complex tasks like TB classification. DenseNet-201's ability to process detailed features from chest X-rays, a critical component in TB diagnosis, enhances the model's classification accuracy. We leveraged this to compare its performance with other CNN architectures, demonstrating its robustness in handling intricate patterns in medical imaging. The inclusion of DenseNet-201, therefore, was not only to provide a comprehensive analysis but also to showcase the effectiveness of different architectural approaches in medical diagnostics.

Hence, we regard the DenseNet201 model as a pivotal component of our model architecture ensemble, along with VGG19 and MobileNetV3 Small, to explore its potential for TB classification. Our findings, highlighting DenseNet-201's performance in TB classification, contribute to the growing body of evidence supporting the use of advanced deep learning techniques in healthcare. This aligns with our paper's aim to explore and validate the integration of cutting-edge machine learning models in medical diagnostics, offering insights into their practical application in e-health systems.

2) VGG 19: In [24], a novel approach employing a tailored VGG19 architecture was utilized to scrutinize chest X-ray images, with a particular focus on distinguishing between normal and pneumonia-afflicted cases. Their model utilized transfer learning in combination with SoftMax and was compared against various prominent deep learning models such as AlexNet, VGG19, and ResNet50. The experimental results unveiled VGG19's superior performance over its counterparts. The VGG19 architecture's use of small, stacked convolution filters significantly enhances its feature extraction capabilities. The small filters enable broader and more complex feature abstraction, crucial for image classification and object detection. VGG19's deep layers excel at extracting high-level features, essential for accurate image classification and object detection. This structure efficiently utilizes contextual information, improving accuracy in tasks like image super-resolution. Additionally, VGG19 effectively balances reducing overfitting and gradient weight updating challenges, maintaining effectiveness across various applications. Furthermore, an Ensemble Feature Selection (EFS) technique was used in [24] to enhance VGG19's diagnostic accuracy. This method fused the hand-engineered features acquired through procedures like Continuous Wavelet Transform (CWT), Discrete Wavelet Transform (DWT), and Gray-Level Co-occurrence Matrix (GLCM) with deep learning features extracted through transfer learning [24]. Both methods are widely used in signal processing and analysis and have different effects on persistence and precision. While the CWT method works more effectively at high frequency, DWT is more successful at the discrete scale level. For this reason, CWT is generally used in applications requiring time-frequency analysis, while DWT is more frequently used in applications requiring multi-level analysis.The remarkable classification accuracy of 97.94% achieved by the VGG19 model, as reported in [24], underscores its clinical utility and its potential applicability for interpreting clinical-grade chest X-rays. Bearing in mind the noteworthy results obtained using VGG19 in chest image analysis as demonstrated in [24], we have opted to integrate the VGG19 model into our research. Its proven proficiency in image classification is anticipated to enrich our study and aid in the precise classification of TB cases. Ensemble Feature Selection (EFS) is a significant method in various data processing and analysis fields, known for its robustness and adaptability in feature selection. Its methodologies, benefits, and practical applications have been explored across different domains, demonstrating its versatility and effectiveness in a wide range of applications, from medical diagnostics to environmental monitoring and cybersecurity, highlighting its importance in modern data analysis and prediction models.

The VGG19 model, known for its robust performance across various metrics, possesses several distinctive characteristics: i) High-Dimensional Feature Extraction: In some applications like blood pressure estimation using PPG signals, the VGG19 model excels in extracting high-dimensional and rich life characteristics, enhancing performance in conjunction with other networks like LSTM [32]. ii) Modification for Specific Applications: For certain tasks, such as detecting Autism Spectrum Disorder from facial images, modifications to the VGG19 model, like altered architecture, attention mechanisms, and the application of transfer learning, have significantly improved its accuracy. These changes enable the model to better capture subtle facial characteristics and reduce overfitting [33]. iii) Transfer Learning Capabilities: The VGG19 model's transfer-learning capabilities are noteworthy, especially in fields like traffic anomaly classification, where it achieves high accuracy and AUC scores, outperforming other methods and previous VGG19 models [34]. iv) Performance Metrics: It achieves impressive performance metrics, such as a testing accuracy of 99.98%, a loss rate of 0.0120, an F1-score of 99.89%, and an area under the ROC of 100% in specific studies [35]. These characteristics collectively contribute to the VGG19 model's robustness and adaptability in various contexts, enabling its consistent outperformance.

3) Mobilenet V3 Small: In [25], the authors performed a comparative analysis of five pre-trained convolutional neural network (CNN) models, specifically, ResNet50, ResNet152V2, DenseNet121, DenseNet201, and MobileNet, to distinguish pneumonia cases from normal instances. The investigation underscored that MobileNet outperformed the other models when operated with a batch size of 16, 64 epochs, and the ADAM optimizer. The model's predictions were further verified using

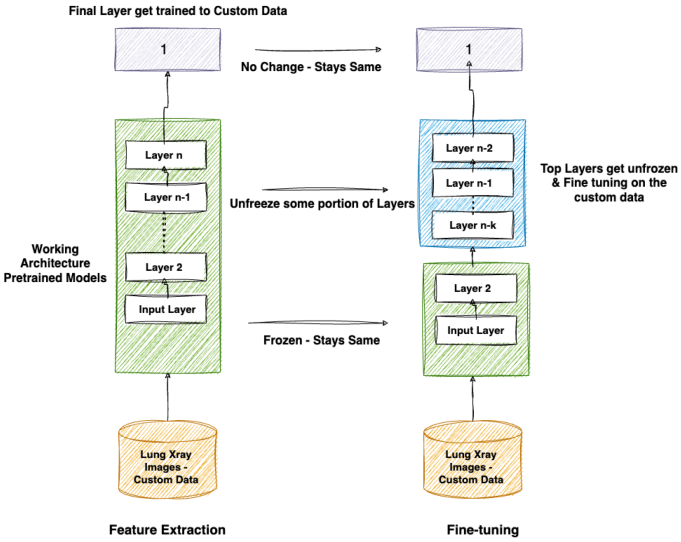

Fig. 1. Process of Feature extraction and fine tuning

public chest radiological images, and the achieved accuracy of the MobileNet model was an impressive 94.23%. Such high accuracy underpins the model's efficacy in aiding the development of effective CNN-based solutions for the preliminary diagnosis of diseases such as COVID-19. Inspired by the promising research findings of [25], we have decided to include MobileNetV3 Small in our study for TB detection. Given the robust performance of the MobileNet architecture in previous research and its ability to deliver high accuracy with lower computational requirements, we anticipate that it will significantly contribute to the success of our TB classification task. **Comparison with Other Models (EfficientNet, ResNet, Lightweight CNNs):** While models like EfficientNet and ResNet are recognized for their efficiency and high performance in various tasks, our choice for Densenet-201, VGG-19, and MobileNetV3 Small was driven by specific considerations relevant to our research context. EfficientNet, known for its scalable architecture, is highly efficient but requires more fine-tuning for specific applications such as medical imaging [36]. ResNet, with its deep residual learning framework, is considered complex for our dataset and task requirements, and its performance in medical image analysis has been outperformed by more specialized architectures [37]. Lightweight deep CNN models often trade off accuracy for efficiency, which is less ideal for the precision required in medical diagnostics [38]. Our chosen models provide a balance between computational efficiency and high accuracy, critical in medical image analysis for TB classification. The extensive literature on Densenet-201, VGG-19, and MobileNetV3 Small, especially in medical imaging contexts, further reinforced our decision [39], [40], [41].

C. Implementation Strategy of Chosen Models

The process of model implementation is divided into two main phases: *Feature Extraction* and *Fine-tuning* as stated in Fig 1, a strategy inspired by an innovative approach.

1) Feature Extraction: In this phase, we utilized the convolutional base of each pre-trained model (DenseNet201, VGG19, and MobileNetV3 Small) as a feature extractor and trained a new classifier on top of it, as suggested in [16]. The pre-trained models are initialized with weights trained on ImageNet. Only the last fully connected layer of each pre-trained model was replaced with a new one, specifically adapted to our binary classification task as shown in Fig 1. Each image is resized to the specific input size required by the pre-trained model (224 $\times$ 224 pixels). The image then passes through the convolutional base of the pre-trained model, which acts as a high-capacity feature extractor, mirroring the methodology [16]. During this phase, the weights of the convolutional base are frozen. Freezing these weights prevents their updates during training, ensuring that the learned pre-trained features are preserved. The ADAM optimizer, known for its efficiency in probability-based models, was chosen for its adaptive learning rate, ideal for managing the binary cross-entropy loss function in our tuberculosis detection model. This optimizer adjusts learning rates based on gradient estimations, leading to robust and efficient training. The training, conducted for 15 epochs with a learning rate of 0.01, benefited from the optimizer's adaptive nature, ensuring preservation of pre-trained features while achieving effective training and better convergence. The training progress is monitored using the TensorBoard callback. Model Checkpoint plays a vital role in the training of ML models by preserving the state of the model at various training stages. Its functions include saving the model weight that yields the best performance, recording the weight for each training epoch, and determining the file format for saving these weights, like HDF5. This feature is critical in ensuring that the most effective model state is retained and can be revisited for optimal performance. The ModelCheckpoint callback is used to save the model weights at the epoch where the validation performance is the best. In addition, weight, bias, and activation values were obtained as a result of quantized and pruned processes for the VGG-19 model. The results show that the learning rate $= 0.01$ should be chosen as the weight value for a more accurate prediction performance of the model. When the bias-variance balance of the model is examined, it is seen that there is no overfit or underfit. Taking "ADAM optimizer" as the activation value indicates that the model has better learning ability.

2) Fine-Tuning: Following the successful training of the top layers, we embark on the fine-tuning phase, an important stage as mentioned in [16]. We unfreeze some of the top layers of the convolutional base and train these layers along with the newly added classifier layers as shown in Fig 1. During the fine-tuning phase of our TB detection model, we unfreeze and train some top layers of the convolutional base alongside the new classifier layers, refining the high-level features for our specific task. We start fine-tuning with a very low learning rate after training the classifier, to control large gradient updates from newly added layers and prevent disruption of the pre-trained features. This approach allows for controlled weight adjustments and enhances model performance while preserving learned features. Fine-tuning must commence with a very low learning rate, typically after the classifier on top of the convolutional base has been trained. This is to avoid large gradient updates caused by the randomly initialized weights, which could disrupt the learned weights in the convolutional base. In our case, we fine-tune the model for an additional 5 epochs with a reduced learning rate of 0.001. The number of layers to unfreeze for fine-tuning depends on the specific model and the task at hand. In our research, we found that unfreezing the last 30% of the layers in each model provided the best results, this insight is

Algorithm 1: Transfer Learning for Tuberculosis Detection.

Input: Train and Test Image Directories: $train\_dir$, $test\_dir$
Output: Trained Binary Classification Model M, Evaluation Metrics $\mathcal{E}$
Variables:
$img\_size \leftarrow (224, 224)$
$batch\_size \leftarrow 32$
$base\_model \leftarrow$ Pre-trained model
$metrics \leftarrow$ Performance metrics
Begin
$\diamond$ *Dataset Preparation*
for each dataset d in $\{train\_dir, test\_dir\}$ **do**
  $D_d \leftarrow$ PrepareImageDataset(d, img_size, $batch\_size$)
end for
$\diamond$ *Model Setup*
$base\_model \leftarrow$ InitializePreTrainedModel()
$M \leftarrow$ AddLayers($base\_model$)
M.Compile(LossFunction, Optimizer, $metrics$)
$\diamond$ *Feature Extraction*
M.Train(D_{train_dir}, D_{test_dir})
$\mathcal{E} \leftarrow M$.Evaluate(D_{test_dir})
$\diamond$ *Fine Tuning*
$base\_model$.UnfreezeLayers()
M.Train(D_{train_dir}, D_{test_dir}, LowLearningRate)
$\mathcal{E} \leftarrow M$.Evaluate(D_{test_dir})
M.SaveModel()
Return M, $\mathcal{E}$
End

aligned with the guidelines provided by [16]. After fine-tuning, the model's performance is evaluated again on the test data, and the model's precision, recall, and F1 score are computed. The trained model and its performance metrics are saved for future use. This strategy of combining transfer learning with fine-tuning allows us to harness the capabilities of pre-trained models and adapt them to our specific task, despite having a relatively small amount of data and computational resources, as validated by the findings in [16]. To provide a comprehensive understanding of our methodology, we refer to Algorithm 1 and Fig. 1. Algorithm 1 delineates a systematic approach to utilizing transfer learning for TB detection, a method that is anticipated to synergize effectively with the emerging IoT and e-health systems.

The algorithm commences with the preparation of the dataset, where images from both training and testing directories are processed and prepared for model training. Following this, a pre-trained model is initialized as the base model, onto which additional layers are added to tailor the model for the specific task of TB detection. The next phase involves feature extraction, where the model is trained and evaluated using the prepared datasets. Subsequently, the model undergoes a fine-tuning process to enhance its performance further. Retraining unfrozen layers of a neural network with a lower learning rate offers several advantages, such as layer-specific adaptation, balanced feature learning, improved mixed-precision quantization, enhanced feature selection and classification, stabilization, error reduction, adaptation post-pruning, and a focus on fine-grained details. This method allows for better adaptation to specific tasks,

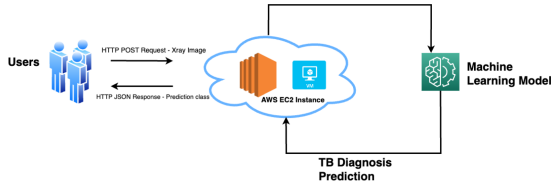

Fig. 2. Illustration of users sending requests to Server Based AWS EC2 Instance running to predict the prediction class of the X-Ray Image

particularly in medical image analysis, by enabling different layers to focus on distinct aspects of the data. It also contributes to a more stable and accurate model, enhances inference time, and maintains accuracy while reducing computational complexity. The fine-tuning involves unfreezing layers in the base model and re-training it with a lower learning rate to achieve a more nuanced understanding of the features in the dataset. Finally, the trained model is stored and the evaluation metrics are returned to demonstrate its performance. The steps to diagnose TB using transfer learning on chest X-ray images are:

- *Data Collection Step:* After obtaining the CT images, we divided them into two: train and test.
- *Transfer Learning Model Selection:* For this purpose, by scanning the literature, models Densenet-201, VGG-19, and Mobilenet-V3-Small, which were proven to be effective in detecting lung-based diseases, were selected.
- *Training the Model:* The model was trained with the train set using pre-trained weights. Attention was paid to bias-variance balance, thus ensuring that the model did not fall into situations such as overfitting and underfitting.
- *Testing the Model:* Test Binary Accuracy (TBA), Test Loss, Test Binary Intersection over Union (IOU), Test Precision, Test Recall, and Test F1 Score values were obtained for the performance evaluation of the model trained using the dataset.

Time Complexity Analysis for Algorithm 1: Aggregating all the components, the total complexity is: $O(2E \cdot (N_{train} + N_{test}) + N_{train} + N_{test})$. This suggests a scaling that is more than linear with the number of images in the training and test datasets, particularly due to the multiplicative factor of epochs E during the training and evaluation phases.

D. Model Deployment Strategies

Server-based and serverless deployments were used in this study to improve solution robustness and scalability.

1) Server-Based Deployment: In this method, the ML models were encapsulated within a FastAPI application. FastAPI is a highly efficient, easy-to-use web framework for building APIs with Python, making it suitable for integration with our ML models[4]. This FastAPI application was then deployed on an Amazon EC2 T3 medium instance. The EC2 instances provide secure and scalable compute capacity in the cloud, allowing our application to efficiently handle incoming HTTP requests[5]. The deployment process and the handling of incoming POST requests are detailed in Algorithm 2 and Fig 2. When a user

[4]https://fastapi.tiangolo.com/
[5]https://aws.amazon.com/ec2/

Algorithm 2: Server-Based TB Detection via FastAPI.

Input: HTTP POST Request R with X-ray Image X
Output: HTTP Response H with Tuberculosis Detection Result Δ
Variables:
$API \leftarrow$ FastAPI application
$EC2\_Instance \leftarrow$ EC2 medium
Begin
$\diamond$ *Deployment Setup*
API.Deploy($EC2\_Instance$)
if R is received **then**
  $X \leftarrow R$.ExtractImage()
  $\Delta \leftarrow API$.Predict(X)
  $H \leftarrow$ CreateResponse(Δ)
  Return H
end if
End

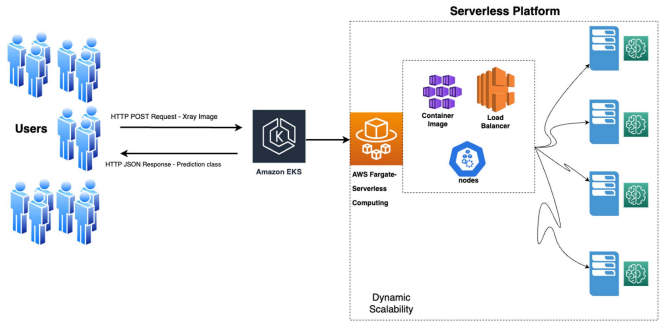

Fig. 3. Illustration of users sending requests to the EKS cluster, with Fargate managing server distribution and facilitating the process of TB prediction using the ML model

sends a POST request (typically when they upload an X-ray image for analysis), the request is processed by the application on the EC2 instance. The image is forwarded to the deployed ML model which processes the image, performs the prediction, and sends back the result in the form of a HTTP response.

The basic concept of traditional server-based deployment, in the case of AWS Elastic Compute Cloud (EC2), involves setting up and managing a server on the cloud to host applications, websites or services. To briefly explain the basic concepts; (i) *Choosing an EC2 Instance:* Users can choose from various instance types depending on their computational, memory, and storage needs. (ii) *Configuring the Instance:* Once an instance type is selected, users can configure the instance settings. (iii) *Setting up the Environment:* After the instance is launched, users can set up the environment based on their requirements. (iv) *Deploying Applications:* Users can then deploy their applications or services onto the EC2 instance. (v) *Security and Maintenance:* Managing a server on EC2 involves ensuring its security and maintenance.

2) Serverless Deployment: In this approach, the FastAPI application that hosts the ML model is deployed in an image using Docker, a tool designed to build, deploy, and run applications using containers. Using Docker enabled efficient management of dependencies and simplified the deployment process to serverless. The deployed image is then deployed using AWS EKS and AWS Fargate. This approach was specially beatifically for our TB detection system as it increased the overall efficiency and responsiveness of the system in a serverless setup. EKS provides a managed Kubernetes service that allows us to automate the deployment, scaling and management of our application. AWS Fargate is a serverless computing engine that eliminates the need to manage servers for containers and thus works with EKS. The process of deploying the FastAPI implementation on Amazon EKS using Fargate serverless computing and the processing of incoming POST requests is described in Algorithm 3 and Fig. 3. Kubernetes Load Balancer service is used to open the application running on the EKS cluster to the Internet. This service is important for making the application accessible over the Internet to handle user-initiated POST requests by efficiently managing incoming network traffic. Load Balancer is used to distribute network traffic to the FastAPI application. This is essential to meet high-volume requests and provide real-time TB detection results. It provided scalability and high availability, which is vital for the robust and responsive performance required in e-health applications. Upon receipt of a user-initiated POST request, the Load Balancer redirects the request to an available container; this container processes the image using the ML model and returns the prediction result as an HTTP response. The user experience is the same with both distribution strategies; The main difference is how the system handles and processes incoming requests internally. The use of Docker for containerizing of FastAPI application was a key decision in this work, directly contributing to the successful deployment and operation of the TB detection model in a serverless environment. To explain this better, it would be useful to go through examples; (i) Docker is used to create a consistent and isolated environment for the FastAPI application and encapsulate the application with all its dependencies (Containerization with Docker), (ii) The containerized application was designed for deployment on serverless platforms like AWS Fargate and AWS EKS, optimizing for scalability and flexibility (facilitating serverless deployment), (iii) The use of Docker ensures that the complex dependencies of the TensorFlow Lite model are managed effectively, leading to fewer deployment-related issues (Advantages in the Context of TB Detection).

3) Deployment in Clinical and Research Settings: Deployment of the system in clinical settings is intended to meet the demands for consistent and potentially high-volume diagnostics. The server-based approach provides a stable and robust platform that guarantees clinical reliability and uninterrupted service essential for patient care. Serverless deployment adapts well to changing patient loads, offering flexible and cost-effective scaling that is vital for a variety of clinical scenarios [42]. In research environments, flexibility and scalability are the focus to cope with changing data sets and experimental conditions. Serverless deployment is particularly advantageous for research purposes due to its scalability and ease of integrating new updates or experimental models. The server-based approach can also be useful for long-term, large-scale research projects that require stable and dedicated computing resources.

The deployment strategies of our TB diagnostic system in both clinical and research settings demonstrate the versatility and adaptability of the system. The design of the system enables it to meet the challenging and variable requirements of these different environments, providing accurate TB detection. As a result, the choice of server-based or serverless deployment strategy is tailored to meet the specific needs of each environment. While

Algorithm 3: Deploying FastAPI on EKS With Fargate.

Input: Docker image I_{docker} of FastAPI application, AWS credentials C_{aws}
Output: Deployed FastAPI application on Amazon EKS with external IP IP_{ext}
$DF \leftarrow$ Define FastAPI and its dependencies in a Dockerfile.
Build I_{docker} from DF and push to DockerHub.
Install AWS CLI and eksctl. Configure with C_{aws}.
$EKSC \leftarrow$ Create EKS cluster with Fargate profile.
$D_{k8s} \leftarrow$ Define Kubernetes Deployment for FastAPI.
Deploy D_{k8s} on $EKSC$. Expose via LoadBalancer to get IP_{ext}.
if POST request R_{post} received **then**
  $Img \leftarrow$ Extract image from R_{post}.
  $Pred \leftarrow$ Predict using ML model with Img.
  Respond with $Pred$.
end if
$CW \leftarrow$ Setup Amazon CloudWatch.
Monitor app and $EKSC$ using CW. Test via POST to IP_{ext}.
End

the server-based approach offers robustness and consistency for clinical applications, the serverless option provides scalability and flexibility, making it ideal for research and development scenarios [43]. This adaptability of delivery strategies enables reliable and effective TB detection, improving the use of this system in a variety of contexts.

4) Time Complexity Analysis of Algorithm 2 & 3: : The time complexity for Algorithm 2 is O(1). Deploying FastAPI on an EC2 instance (Deployment Setup) is a one-time process and contributes to a constant overhead. Likewise, the time complexity for Algorithm 3 is O(1). This is because you need to set up the EKS cluster, define the Kubernetes deployment, etc. System configuration and deployment processes such as contribute to a constant overhead. Both algorithms possess a constant overhead due to setup processes. The variable component of their complexities is linear, attributed to the processing of images or POST requests, respectively. Thus, in the context of time complexity, both algorithms scale linearly, but with different input factors.

IV. Performance Evaluation

This section evaluates the performance of ML models and deployment strategies for TB detection system. The system uses ML models to analyze X-ray images and predict the presence of TB. We trained Densenet-201, VGG19, and Mobilenet-V3-Small and deployed the system in two environments: a server-based deployment using an EC2 T3 medium instance and AWS EKS and Fargate based a serverless settings.

A. Evaluation Metrics

The choice of metrics for evaluating the ML models was guided by the nature of the problem and the requirements of the system. Accuracy, loss, IOU (Intersection over Union), precision, recall, and F1 score were chosen as they provide a comprehensive view of the model's performance[6]. Accuracy measures the proportion of correct predictions, while loss quantifies the difference between the predicted and actual values. IOU is a measure of overlap between the predicted and actual areas, which is particularly relevant for image analysis tasks. Precision, recall, and F1 score provide insights into the model's performance in terms of false positives and false negatives, which are crucial in medical diagnosis systems where both false positives and false negatives have significant implications [44]. The IoU metric is a popular evaluation metric used in various image processing and computer vision tasks, especially in object detection and segmentation. The IoU metric is employed to assess the overlap between the predicted area and the actual ground-truth area. It measures the accuracy of a model's predictions by calculating the overlap between the predicted area and the actual ground-truth area. A higher IoU score indicates a greater accuracy. Unlike simpler accuracy metrics, IoU considers both true positives and false positives, making it more robust, especially in datasets with extensive background space. IoU offers a standardized way to compare the performance of different models or algorithms on the same dataset, aiding in benchmarks and competitions. Applicable to various tasks like object detection, image segmentation, and tracking, IoU is versatile in assessing spatial accuracy of models. It balances the aspects of precision and recall, providing a comprehensive performance metric. For evaluating the deployment strategies, we considered the average response rate, error percentage, and throughput under different loads[7]. These metrics were chosen as they reflect the system's ability to handle multiple concurrent requests, which is a key requirement for a practical, real-world application. The average response rate indicates the system's speed, while the error percentage provides an indication of its reliability. Throughput, measured as the number of requests handled per unit of time, gives an idea of the system's capacity[8]. By comparing these metrics across different models and deployment strategies, we aim to identify the most effective combination for our tuberculosis detection system. This will enable us to optimize the system for better performance, reliability, and capacity, thereby improving its utility in real-world applications.

TABLE II
Performance Comparison of ML Models

Metrics	DenseNet-201	VGG19	MobileNet-V3-Small
TBA	0.7826	0.8633	0.826
Test Loss	0.6023	0.4212	0.3532
Test Binary IOU	0.6425	0.7595	0.7036
Test Precision	0.82191	0.875	0.8292
Test Recall	0.7317	0.8536	0.83
Test F1 Score	0.7741	0.8641	0.83

B. Performance Comparison of ML Models

Table II presents the comparative analysis of ML models: DenseNet-201, VGG19, and MobileNet-V3-Small. We evaluated the models based on several performance metrics such as Test Binary Accuracy (TBA), Test Loss, Test Binary IOU, Test Precision, Test Recall, and Test F1 Score. Upon examination of the data, it is evident that the VGG19 model consistently

[6]https://ml-compiled.readthedocs.io/en/latest/metrics.html

[7]https://pflb.us/blog/load-testing-metrics/

[8]https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testing https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testing

outperforms the other two models across nearly all metrics. The model boasts the highest Test Binary Accuracy at 86.33%, which signifies a greater proportion of correct predictions. Furthermore, it also excels in terms of Test Precision, at 87.5%, indicating a lower rate of false-positive results. In the context of Test Recall and Test F1 Score, VGG19 leads with scores of 85.36% and 86.41% respectively, demonstrating a balanced performance between precision and sensitivity. DenseNet-201 showed lower performance compared to VGG19 in our TB classification study, recording the highest Test Loss at 60.23%. The architectural design differences between the models contributed to this outcome. VGG19, with its simple and deep stack of small filters, is more effective at capturing hierarchical features, leading to its higher performance, stability, and reliability. DenseNet-201's structure, though efficient in some scenarios, was less aligned with the demands of tuberculosis detection from chest X-rays. The VGG19 model's robustness and reliability are attributed to its architectural design, effective transfer learning approach, and proven performance across diverse applications. Its simple, deep architecture with small filters in convolutional layers enhances hierarchical feature capture, contributing to its high performance. The model's adaptability and consistency are further demonstrated in various applications, from crack detection in infrastructure to grape bunch segmentation in natural images, showing its ability to maintain stability and reliability across different tasks and data types.

On the other hand, both DenseNet-201 and MobileNet-V3-Small fail to match the performance of the VGG19 model. Particularly, DenseNet-201 exhibits the lowest scores across all the metrics, including the highest Test Loss of 60.23%. This disparity in performance can be attributed to several factors intrinsic to the architecture of each model. DenseNet-201, while effective in feature preservation and reuse, can be computationally intensive due to its dense connectivity. This may lead to challenges in capturing more abstract features in complex medical images like chest X-rays, crucial for accurate TB classification. MobileNet-V3-Small, designed for efficiency and speed, may compromise the depth and breadth of feature extraction needed for TB detection. In contrast, VGG19's deep architecture allows it to excel in capturing a wide range of features from medical images, outperforming the other models in our study by effectively learning and differentiating subtle features indicative of tuberculosis from chest X-rays. Although MobileNet-V3-Small shows competitive scores in some areas, such as Test Precision, its overall performance is still not on par with VGG19. Robustness is important while choosing a deployment model. In this regard, all metrics demonstrate VGG19 to be high-performing, stable, and dependable. This reliability can be attributed to the architecture of the VGG19 model, which employs a simple and deep stack of small filters in its convolutional layers, enhancing the model's ability to capture hierarchical features effectively. Based on the performance indicators and the architectural benefits, we are selecting VGG19 as the model of choice for deployment.

C. Performance Comparison of Deployment Strategies

Each cloud service deployment strategy has pros and cons. These techniques meet various use cases, requirements, and scalability demands. Herein, we compare two prevalent deployment strategies: a traditional EC2-based approach utilizing a t3.medium server instance and a more modern, serverless approach using Fargate and EKS. These analyses and comparisons will offer a clearer understanding of the behavior of these strategies under different loads, potentially guiding deployment decisions for similar scenarios. The tests were conducted using Apache JMeter, a renowned performance testing tool. For our experiment, we subjected both deployment strategies to a range of concurrent user requests, specifically, starting from 20 and scaling up to 500. To ensure the results were not affected by initial cold starts or sudden spikes, a ramp-up period of 100 seconds was used, with each simulation running in a single loop. The results of this experiment are detailed in this section. Table III presents the performance metrics of the EC2 t3.medium server under varying levels of concurrent user requests, while Table IV depicts similar metrics for a serverless architecture using Fargate and EKS. **Several key observations can be drawn from the data:**

TABLE III
PERFORMANCE METRICS OF AN EC2 T3.MEDIUM SERVER UNDER VARYING LEVELS OF CONCURRENT USER REQUESTS OVER A 100-SECOND RAMP-UP PERIOD

Samples	Avg Response Rate(ms)	Error %	Throughput
20	4691	0	12.0 /min
40	47 724	0	15.7 /min
60	123 333	0	15.4 /min
80	190 376	0	15.4 /min
100	269 971	0	15.1 /min
200	653 522	0	14.3 /min
300	922 305	0	15.2 /min
400	1 255 053	0.50	15.4 /min

TABLE IV
PERFORMANCE METRICS OF SERVERLESS FARGATE AND EKS UNDER DIFFERENT LEVELS OF CONCURRENT USER REQUESTS

Samples	Avg Response Rate(ms)	Error %	Throughput
20	8026	0	11.7 /min
40	50 905	0	14.6 /min
60	133 901	0	14.2 /min
80	239 146	0	12.9 /min
100	272 432	0	14.8 /min
200	636 805	8.00	14.8 /min
300	453 248	34.67	21.3 /min
400	929 529	19.50	18.4 /min
500	978 658	34.40	20.2 /min

1) Initial Performance: For lower user loads, both deployment strategies manage to handle requests efficiently with zero errors. The traditional EC2 instance has a slightly better throughput at 20 concurrent users, but as the user load increases, the serverless approach tends to cope better.

2) Average Response Rate: An apparent increment in the average response rate is observed for the serverless architecture when the number of users jumps from 100 to 200. Conversely, the EC2 instance demonstrates a more consistent increase.

3) Error Rate: At higher levels of user concurrency, the serverless architecture, surprisingly, showcases non-zero error percentages. Specifically, at 200 concurrent users and beyond, errors emerge, peaking at a significant 34.67% for 300 users. This behavior suggests that while the serverless approach is highly scalable, it may experience difficulties under sudden and intense load surges. Serverless computing is known for its high scalability, which is characterized by automatic scaling to match demand, the use of microservices allowing independent scaling of application components, and cost-effectiveness with payment models based on actual resource usage [43]. However, the role

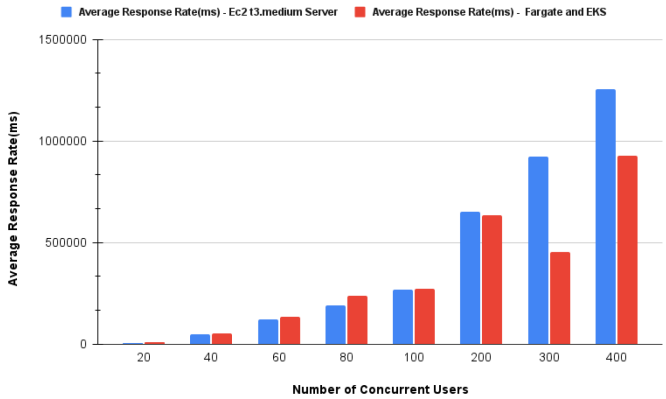

Fig. 4. Comparison of Average Response Rate for Server-based (EC2) and Serverless (Fargate and EKS) Architectures with Varying Concurrent Users.

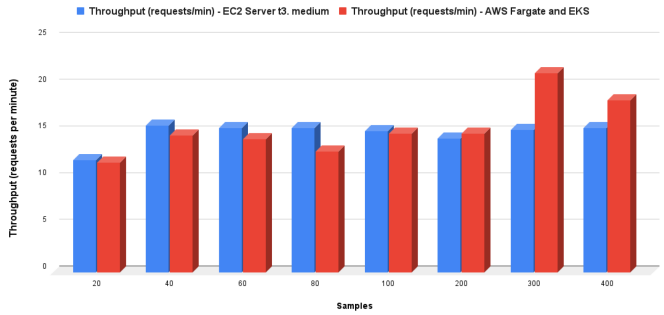

Fig. 5. Comparison of Throughput (requests per minute) for Server-based (EC2) and Serverless (Fargate and EKS) Architectures with Varying Concurrent Users.

of error rates in scalability cannot be overlooked. Low error rates lead to more efficient resource utilization, whereas high error rates may necessitate additional resources to manage failed executions. Serverless platforms use error rates to manage the health of functions, allocating more resources when high error rates are detected. Analyzing error rates also provides feedback for optimizing applications, thereby improving scalability. In conclusion, the scalability of serverless methods is greatly influenced by error rates. Efficient management of these rates is crucial for maintaining optimal scalability and performance in serverless architectures.

4) Throughput: It is a measure of how many units of information a system can process in a given amount of time, and doesn't exhibit a consistent trend for either architecture. While the serverless approach has slightly decreased throughput at 80 users, it bounces back and surpasses the EC2 instance at higher loads. AWS Fargate's integration with EKS enhances throughput by offering innovative cloud container management, particularly beneficial for scientific computing. This setup provides cost-effective performance, elasticity, scalability, and reduced delays. Fargate's on-demand capacity is crucial for dynamic scaling, optimizing resource utilization and ensuring consistent throughput across varying workloads. The simplification in managing Kubernetes clusters, by eliminating server provisioning and management, indirectly boosts throughput. Additionally, optimized container orchestration and network routing in Fargate-EKS integration improve network performance, which is essential for high throughput in distributed applications.

Fig. 4 presents a visual comparison between the average response rate of server-based deployments, using an EC2 instance, and serverless configurations utilizing Fargate and EKS, benchmarked against varying levels of concurrent user requests. Each column in the chart corresponds to the average response rate for a specific number of concurrent users. Initially, at lower user counts, the height of the columns for both deployment strategies are quite similar, signifying nearly equivalent performance metrics under light loads. As the concurrent user number rises, distinct differences in the column heights begin to emerge. The EC2-based deployment columns grow in a more steady and uniform manner, illustrating the predictable nature of traditional server-based architectures. It shows how, with the increase in load, the server's response time exhibits a gradual and linear degradation.

In contrast, columns representing the serverless architectures of Fargate and EKS exhibit a more variable growth pattern. Up to a certain threshold of users, the response rates are competitive, possibly even outperforming the EC2 instance. This resonates with the on-the-fly scalability of serverless models. However, at the upper echelons of user counts, the column heights for the serverless configurations become notably taller, indicating increased response times. An increased response time, especially in serverless configurations like Fargate and EKS, indicates a potential overhead related to cold starts or the time required to provision additional resources in real-time. This variability in response times represents a trade-off between the steady performance of traditional server systems and the dynamic but sometimes unpredictable nature of serverless setups. The average response rate is a critical metric reflecting the system's speed. A higher response rate implies a slower system, which can impact the overall efficiency of the application, particularly in real-world scenarios where timely processing is essential. Moreover, the error percentage, which indicates the reliability of the system, can also be affected by increased response times. In our system, these aspects are crucial for handling multiple concurrent requests efficiently, a key requirement for practical applications like our tuberculosis detection system. This could be symptomatic of the overheads related to cold starts or the time taken to provision additional resources in real time In essence, Fig. 4 effectively captures the balance between the steady performance of traditional server systems and the dynamic, but occasionally unpredictable, nature of serverless setups. The decision between the two would hinge on specific use-case requirements, traffic predictability, and tolerance for variability in response times. In Fig. 5, the comparative throughput performances of two deployment strategies–EC2 Server t3.medium and AWS Fargate combined with EKS–are visualized across varying concurrent user loads. For the lower user count, specifically up to 80 concurrent users, both deployment strategies exhibit relatively close throughputs, with EC2 marginally outperforming Fargate and EKS in most cases. However, as we transition to higher user loads, the gap in performance becomes more apparent. Notably, at 300 concurrent users, the AWS Fargate and EKS combination achieves a significant jump, reaching a throughput of 21.3 requests/min, which surpasses the EC2's consistent performance of around 15.4 requests/min. This suggests that, under higher loads, the serverless architecture of Fargate combined with EKS might offer better scalability and responsiveness compared to the t3.medium instance of EC2.

V. Conclusions and Future Work

TB remains a formidable global health challenge. To address this, our paper embarked on a rigorous exploration of various CNN architectures, capitalizing on the power of transfer learning to predict TB presence in the lungs. The fusion of AI and ML was instrumental in this endeavor, illuminating the latent potential of technology in healthcare diagnostics. In our study, the VGG19 model stood out as the top performer based on test data results. Still, Densenet-201 and Mobilenet-V3-Small also delivered impressive outcomes, highlighting the crucial role of picking the right model for specific diagnostic needs. We adopted serverless computing, specifically AWS Fargate, and EKS, due to its unmatched flexibility, cost-effectiveness, and on-the-fly resource allocation, making it the go-to choice for real-time medical applications. Traditional servers, like EC2 t3.medium, while consistent, cannot match the dynamic scalability of serverless options. However, it's worth noting that serverless solutions can sometimes experience "cold start latency", a minor hiccup in an otherwise superior solution[9]. Our observational metrics revealed a parity in throughput between both deployment paradigms at lower user concurrency. However, the scalability acumen of serverless computing came to the fore with rising user requests, distinctly outshining traditional servers around the 300-user threshold. Caution, however, is warranted. Our serverless deployments, while agile, began registering heightened error rates beyond 200 concurrent interactions, hinting at possible vulnerabilities when grappling with abrupt traffic surges.

Building upon the conclusions, we identify several promising avenues for future research: i) *Exploring Advanced Architectures:* Delving deeper into newer or hybrid neural network architectures might yield even better diagnostic accuracies. To improve this paper in future studies, chest X-ray images from different hospitals can be studied for a federated learning environment. Additionally, using more advanced transfer learning models in the future may further increase the prediction rate on TB. ii) *Real-time Diagnostics:* Extend the model implementations for real-time analysis, catering to continuous data streams from medical diagnostics equipment. iii) *Optimizing Deployment Strategies:* A focused study on fine-tuning both server and serverless deployments could help in further reducing latencies and improving resource utilization. iv) *Incorporating Diverse Data Points:* Broadening the dataset to include parameters like patient history and demographics may improve the predictive accuracy of the models. v) *Disease Spectrum Extension:* The methodology's success for TB suggests its applicability to other diseases. Extending this to other medical conditions could be groundbreaking. vi) *Comprehensive Cost-Benefit Analysis:* As cloud and serverless technologies evolve, a detailed analysis considering cost, performance, and reliability could guide institutions in deployment decisions. vii) *Serverless Paradigm based Shortcomings:* In addition to the previously mentioned advantages, serverless platforms also bring with them some disadvantages that need to be solved, such as cold start latency, error rates at high concurrency, and traditional VM limitations. Efforts to resolve these concerns will increase users' trust in the system.

[9]https://azure.microsoft.com/en-us/blog/understanding-serverless-cold-start/

Software Availability

It has been released as open-source software. The implementation code with experiment scripts and results can be found in the GitHub repository: https://github.com/Subramaniam-dot/CNN-Architectures-and-Deployment-Models-for-Tuberculosis-Detection-Using-Serverless-Computinge-healthcarefaas

References

[1] M. Singh et al., "Evolution of machine learning in tuberculosis diagnosis: A review of deep learning-based medical applications," *Electronics*, vol. 11, no. 17, 2022, Art. no. 2634.

[2] R. Kariapper, M. S. Razeeth, P. Pirapuraj, and A. Nafrees, "RFID based smart healthcare system: A survey analysis," *Test Eng. Manage.*, vol. 83, pp. 4615–4621, 2020.

[3] C. Mansoor, A. C. M. Nafrees, S. A. Asra, and M. I. Jahan, "A new paradigm for healthcare system using emerging technologies," in *Proc. Int. Conf. Comput. Eng. Technol.*, 2022, pp. 311–322.

[4] A. C. M. Nafrees et al., "Smart technologies to reduce the spreading of COVID-19: A survey study," in *Proc. Int. Conf. Intell. Vis. Comput.*, 2021, pp. 250–265.

[5] S. Razeeth et al., "Understanding the identity of a COVID-19 suspect or victim through the use of Google glass," in *Proc. 2nd Int. Conf. Proc. Building Sustain. Future Through Technol. Trans.*, Sri Lanka, 2022, pp. 24–33.

[6] A. C. M. Nafrees, A. M. A. Sujah, and C. Mansoor, "Smart cities: Emerging technologies and potential solutions to the cyber security threads," in *Proc. 5th Int. Conf. Elect., Electron., Commun., Comput. Technol. Optim. Techn.*, 2021, pp. 220–228.

[7] A. Rejeb et al., "The Internet of Things (IoT) in healthcare: Taking stock and moving forward," *Internet Things*, vol. 22, 2023, Art. no. 100721.

[8] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–21.

[9] M. Abernot et al., "Digital implementation of oscillatory neural network for image recognition applications," *Front. Neurosci.*, vol. 15, 2021, Art. no. 713054.

[10] O. Kaziha, A. Jarndal, and T. Bonny, "Genetic algorithm augmented convolutional neural network for image recognition applications," in *Proc. Int. Conf. Commun., Comput., Cybersecurity, Informat.*, 2020, pp. 1–5.

[11] J. Tanabe et al., "18.2 a 1.9 TOPS and 564GOPS/W heterogeneous multicore SoC with color-based object classification accelerator for image-recognition applications," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2015, pp. 1–3.

[12] Y. Zhang et al., "Sequence-to-sequence domain adaptation network for robust text image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2735–2744.

[13] J. Su et al., "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf.*, 2018, pp. 664–669.

[14] M. H. Modarres et al., "Neural network for nanoscience scanning electron microscope image recognition," *Sci. Rep.*, vol. 7, no. 1, 2017, Art. no. 13282.

[15] J. Xiong, D. Yu, S. Liu, L. Shu, X. Wang, and Z. Liu, "A review of plant phenotypic image recognition technology based on deep learning," *Electronics*, vol. 10, no. 1, 2021, Art. no. 81.

[16] S. Showkat and S. Qureshi, "Efficacy of transfer learning-based resnet models in chest X-ray image classification for detecting COVID-19 pneumonia," *Chemometrics Intell. Lab. Syst.*, vol. 224, pp. 104534–104534, 2022.

[17] S. Jaeger et al., "Two public chest X-ray datasets for computer-aided screening of pulmonary diseases," *PubMed*, vol. 4, no. 6, pp. 475–477, 2014.

[18] S. S. Gill et al., "Modern computing: Vision and challenges," *Telematics Informat. Rep.*, vol. 13, 2024, Art. no. 100116.

[19] S. Ravimohan, H. Kornfeld, D. Weissman, and G. Bisson, "Tuberculosis and lung damage: From epidemiology to pathophysiology," *Eur. Respir. Rev.*, vol. 27, no. 147, 2018, Art. no. 170077.

[20] C. Qin, D. Yao, Y. Shi, and Z. Song, "Computer-aided detection in chest radiography based on artificial intelligence: A survey," *Biomed. Eng. Online*, vol. 17, no. 1, 2018, Art. no. 113.

[21] F. Carvalho, D. R. Silva, and M. P. Dalcolmo, "Tuberculosis: Where are we?," *J. Brasileiro De Pneumologia*, vol. 44, no. 2, pp. 82–82, 2018.
[22] A. Shelke et al., "Chest X-ray classification using deep learning for automated COVID-19 screening," *SN Comput. Sci.*, vol. 2, no. 4, 2021, Art. no. 300.
[23] V. S. K. Tangudu, J. Kakarla, and I. B. Venkateswarlu, "COVID-19 detection from chest X-ray using mobilenet and residual separable convolution block," *Soft Comput.*, vol. 26, no. 5, pp. 2197–2208, 2022.
[24] N. Dey, Y. Zhang, V. Rajinikanth, R. Pugalenthi, and M. Raja, "Customized VGG19 architecture for pneumonia detection in chest X-rays," *Pattern Recognit. Lett.*, vol. 143, pp. 67–74, 2021.
[25] M. Saleh et al., "Detection of pneumonia from chest X-ray images utilizing mobilenet model," *Healthcare*, vol. 11, no. 11, pp. 1561–1561, 2023.
[26] E. Showkatian, M. Salehi, H. Ghaffari, R. Reiazi, and N. Sadighi, "Deep learning-based automatic detection of tuberculosis disease in chest X-ray images," *Polish J. Radiol.*, vol. 87, no. 1, pp. 118–124, 2022.
[27] A. Jaiswal, N. Gianchandani, D. Singh, V. Kumar, and M. Kaur, "Classification of the COVID-19 infected patients using densenet201 based deep transfer learning," *J. Biomol. Struct. Dyn.*, vol. 39, no. 15, pp. 5682–5689, 2020.
[28] S. Jaeger et al., "Automatic tuberculosis screening using chest radiographs," *IEEE Trans. Med. Imag.*, vol. 33, no. 2, pp. 233–245, Feb. 2014.
[29] S. Candemir et al., "Lung segmentation in chest radiographs using anatomical atlases with nonrigid registration," *IEEE Trans. Med. Imag.*, vol. 33, no. 2, pp. 577–590, Feb. 2014.
[30] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[31] M. J. Sheller et al., "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Proc. Brainlesion: Glioma, Mult. Sclerosis, Stroke Traumatic Brain Injuries: 4th Int. Workshop*, 2018, pp. 92–104.
[32] D. Bhattacharjee, "Cuff-less blood pressure estimation from electrocardiogram and photoplethysmography based on VGG19-LSTM network," in *Proc. Comput. Methods Med. Health Care: Proc. Workshop*, IOS Press, 2021, Art. no. 33.
[33] R. Chandra et al., "Autism spectrum disorder detection using autistic image dataset," in *Proc. 10th Int. Conf. Elect. Eng., Comput. Sci. Informat.*, 2023, pp. 54–59.
[34] S. Bouhsissin, N. Sael, and F. Benabbou, "Enhanced VGG19 model for accident detection and classification from video," in *Proc. Int. Conf. Digit. Age Technological Adv. Sustain. Develop.*, 2021, pp. 39–46.
[35] S. Mohsen et al., "Brain tumor classification using hybrid single image super-resolution technique with ResNext101_32x8d and VGG19 pre-trained models," *IEEE Access*, vol. 11, pp. 55582–55595, 2023.
[36] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[38] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[39] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–14.
[41] A. Howard et al., "Searching for mobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
[42] M. Golec, S. S. Gill, A. K. Parlikad, and S. Uhlig, "HealthFaaS: AI based smart healthcare system for heart patients using serverless computing," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 18469–18476, Nov. 2023.
[43] M. Golec et al., "ATOM: AI-powered sustainable resource management for serverless edge computing environments," *IEEE Trans. Sustain. Comput.*, early access, Dec. 29, 2023, doi: 10.1109/TSUSC.2023.3348157.
[44] S. Hicks et al., "On evaluation metrics for medical applications of artificial intelligence," *Sci. Rep.*, vol. 12, no. 1, 2022, Art. no. 5979.