



Lung Mass Identification Using Tiny Deep Learning Based on Lightweight MobileNet and Raspberry Pi 5 for Low Source Medical Diagnostic

Yasir Salam Abdulhafoor^{*}, Auns Qusai Al-Neami^{}, Ahmed Faeq Hussein^{}

Biomedical Engineering Department, College of Engineering, Al-Nahrain University, Baghdad 10011, Iraq

Corresponding Author Email: yasir.salam.phd2023@ced.nahrainuniv.edu.iq

Copyright: ©2025 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/isi.300714>

ABSTRACT

Received: 17 June 2025

Revised: 20 July 2025

Accepted: 25 July 2025

Available online: 31 July 2025

Keywords:

tiny deep learning, lung mass, MobileNetV2, resource-constrained devices, Raspberry Pi 5, medical imaging, edge AI

Medical imaging analysis has greatly benefited from deep learning, especially Convolutional Neural Networks (CNNs). However, their enormous parameter sizes and high computational cost restrict their use on low-resource systems. This paper suggests a small deep learning solution for resource-constrained contexts that uses a lightweight CNN, MobileNetV2, deployed on a Raspberry Pi 5 to enable lung mass identification from chest X-ray (CXR) pictures. A total of 2,322 NIH CXR pictures tagged as normal or mass were used to assess two iterations of the model: pretrained and trained from scratch. With just 3.4 million parameters and 300 million FLOPs, the pretrained MobileNetV2 obtained a validation accuracy of 95.25%, test accuracy of 89.9%, precision of 91.43%, and F1 score of 90.14%. A validation accuracy of 91.03%, test accuracy of 85.06%, precision of 88.24%, and F1 score of 85.21% were attained by the scratch-trained version. The results show that it is possible to implement precise CNN-based medical diagnostics on inexpensive, low-power devices, which could increase access to AI-assisted healthcare in underprivileged areas. This study demonstrates the feasibility of a lightweight deep learning model for real-time lung mass detection in resource-constrained medical settings by presenting its end-to-end deployment on the Raspberry Pi 5, from training to on-device inference.

1. INTRODUCTION

An abnormal spot, tumor, nodule, or area in the lungs that is more than 3 cm (1.5 inch) in size is defined as a "lung mass" [1]. This mass can be caused by lung infection and diseases or may be fibrosis that resulted from tuberculosis and Covid19 disease, almost the mass may be tumors and these tumors may be benign such hamartoma, lipoid, lipoma and pneumonia. But the commonest cause of a pulmonary mass is lung cancer which is malignant tumors. One of the most common cancers is lung cancer, yearly, over 225,000 cases, 150,000 deaths, and \$12 billion in medical expenses occurred in the United States [2]. Lung cancer is regarded as one of the most deadly malignancies; in the United States, just 17% of patients survive five years after receiving a diagnosis, and the survival rate is worse in developing nations. How the cancer it has metastasized, it refers to cancer's stage. Cancers that localized to the lungs refer to Stages 1 and 2 and cancers that have spread to other organs refer to latter stages. Biopsies and imaging are the current diagnostic methods, such as CT scans. Early discovery of lung cancer significantly increases the chances of survival; yet, because there are fewer symptoms, it is more challenging to diagnose lung cancer in its early stages [3]. The two most common and distinct forms of lung cancer are non-small cell lung cancer (NSCLC) and small cell lung cancer (SCLC) [2]. Regardless of gender, small cell lung cancer typically develops in chronic smokers. NSCLC, a unique word among several forms of lung cancer, is the most

prevalent type of the disease overall. NSCLC will mostly contain large cell carcinoma, adenocarcinoma, and squamous cell carcinoma. Computer-aided techniques are mostly used for lung cancer identification in order to increase accuracy and lower costs, which facilitates a quicker and more effective recovery from the illness. Deep learning techniques can address a variety of issues, including speech recognition, object recognition, and natural language processing.

CNN is extensively utilized since it is the primary deep learning tool for extraction; multilayer convolution and max pooling techniques are employed to extract features. CNN extracts a number of features that are present at the buried layer. The CNN network primarily consists of two convolutional layers, namely a pooling layer and a drop out layer for segmentation [4]. Pooling reduces the size of the representation and the network estimation. Additionally, max pooling is employed, and the subsequent layer, known as the dropout layer, keeps the neural network from overfitting by ignoring the randomly chosen neurons during training.

Well-known deep neural networks (NNs) usually have tens of millions of parameters, and the most sophisticated models use a lot of memory. Moreover, the implementation of the most sophisticated model on portable devices is challenging due to the high-performance hardware resources needed for deep neural networks [5].

Aside from the technological constraints, ML and CNN models have demonstrated promising results in previous studies; yet, certain problems remain. The computational

expense of these models is a major disadvantage; their millions of parameters necessitate fine tuning and take more time, making them inappropriate for devices with low resources. Super computers or special hardware having super features that related to (CPU, GPU, memory, power) are required to deploy heavy weight CNN models and these features need a high cost besides these special computers or traditional hardware consume a large amount of power and energy to implement image classification.

Researchers recommended using Tiny ML and Tiny DL approaches as a result. They created CNN models that are lightweight, resource-constrained, and useful for a variety of medical applications, including the diagnosis of various and deadly diseases like cancer, particularly lung tumor/cancer prediction.

The significant interest in research and industry has sparked by convergence of machine learning and Internet of Things (IoT) because the processing of local data is enabled by embedded hardware and interacted with their environment in an intelligent and automated way. This intersection creates the new field of TinyML, a term that Han and Siebert first used in 2019 [6]; see Figure 1. Designing small and effective neural network models and deployment strategies for energy-constrained devices, like those found in the Internet of Things (IoT), are the main goals of TinyML. Devices can sense and analyze local data like sound, temperature, pressure, and motion right at the source thanks to Microcontroller units (MCUs) and Micro-Electro-Mechanical Systems (MEMS), which are commonly used in Internet of Things systems. Common uses include gesture recognition, event counting, predictive maintenance, keyword identification in voice assistants, and integration into consumer goods, including augmented reality glasses, smartphones, smart watches, smart appliances, and remote controls. The ability of powerful hardware, especially graphics processing units (GPUs), to handle the high computational needs of deep learning has been a major factor in its quick development. On the other hand, edge deployment has proven more difficult because to the delayed acceptance of deep learning on systems with minimal resources, like microcontrollers. However, the rising variety of embedded devices and the potential of deep learning present exciting prospects for industry and scholars alike [7].

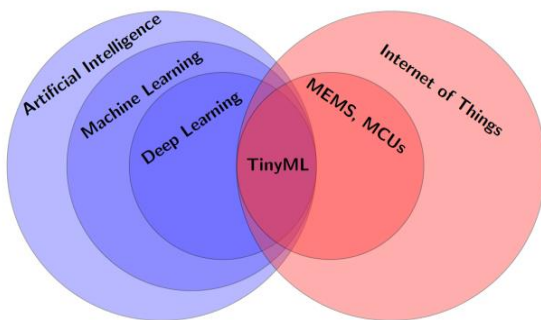


Figure 1. TinyML as the nexus of embedded systems and artificial intelligence [7]

SqueezeNet [8], MobileNet V1, V2, and V3 [9, 10], or EfficientNet [10] are regarded as new, effective deep learning models that have arisen. They are capable of operating on mobile devices with one to five million parameters. When compared to AlexNet [10]. These new models can reduce the model size by up to 5-10 times while maintaining the same

level of performance. In general, the size of the models is at least 10^6 .

Responsive wearable applications and smart embedded are enabled by on-device deep learning in which, image, audio and other data of sensor from the user for to do specific task such as human activity recognition (HAR) are processed by the studies [11, 12], sensing of audio [12], tracking of hand [13], recognition of handwriting [14], classification of image [15], and enhancement of images [16]. Microcontroller units (MCUs) are often supplying the power to embedded systems and wearable internally. Microcontroller units are single-chip devices that usually have a single-core processor (like the ARM M-series), on-chip persistent Flash, and temporary SRAM memory. They are also often found close to additional application-specific peripherals like sensors and radio microphones. When designing an MCU, cost and power efficiency should be considered. This architecture can reduce onboard memory and computing resources. For many users, getting beyond microcontrollers' resource limitations is not a small matter. Even when compared to microcomputers like Raspberry Pi, MCUs have instructions that are a lot slower (GHz vs. 100s MHz) and have less SRAM and storage (GBs vs. 100s KB). As a result, it is very difficult to run fully functional neural networks on MCUs. This study accomplishes complete end-to-end deployment on the recently announced Raspberry Pi 5, in contrast to research that only assess models in high-computational contexts. The suggested system demonstrates the viability of providing portable, affordable, and self-contained diagnostic tools for clinical application in resource-constrained environments by covering the entire pipeline, from model training to direct on-device inference.

2. RELATED WORK

In this section, some related works that focused on light weight MobileNet model and tiny deep-learning techniques and algorithms that were used for lung cancer, lung diseases and other diseases classification.

Ghosh et al. [17] suggested automatic categorization of lung nodules in CT images using DL models, such as CNN, VGG16, and effective MobileNet models, to increase diagnostic speed and accuracy. The findings showed that employing MobileNet with transfer learning achieved high accuracy of 99.11%. Mothkur and Veerappa [18], to detect early stages of lung cancer in CT images, researchers employed vanilla CNN and two lightweight deep neural networks, MobileNet and SqueezeNet. The results indicate that MobileNet had the highest accuracy while SqueezeNet had the shortest processing time.

Mohammed [19] employed light weight MobileNet model & CNN model with energy valley optimization for hyperparameter tuning to classify pulmonary diseases in Xray images, the results show that using MobileNet model with energy valley optimization improved accuracy to reach 85.91%. Rahman et al. [20], to predict lung cancer in CT scans, the VGG8 model, Inception v3, and lightweight MobileNet model were recommended and used by authors, the results show that MobileNet model has a better performance while splitting CT images than inception v3 and VGG8 mdodels. Souid et al. [21] proposed modified MobileNetV2 model to classify lung diseases using NIH X-ray chest dataset images, the results indicate that accuracy reached to 90% and the area

under receiver operating characteristic curve AUC was 0.811%. Mengistie and Kumar [22], in order to classify and predict lung diseases using X-ray images, the authors suggested using VGG19, ResNet50, DensNet201, and the lightweight model MobileNetV2. The results indicate that MobileNetV2 performs better, with an accuracy of 98.49%, while the remaining models, VGG19, ResNet50, and DensNet201, achieved accuracy of 97.43%, 94.64%, and 98.05%, respectively. The MobileNetV2 model also required less training time than the other models. Yaqoob et al. [23] employed light weight MobileNetV2 and SqueezeNet models to classify Covid19 using X ray chest images, the results show that the MobileNetV2 achieved better performance with F1-score of 97.06%, accuracy of 97%, specificity of 95%, area under the curve of 98.93%, precision of 95.19%, recall of 100%. Kumar et al. [24] employed deep learning technique to detect breast cancer in histopathological images, they deployed light weight MobiHisNet model on Raspberry pi2, they achieved good accuracy, lesser computational memory cost and requirement. Biswas and Barma [25] achieved 98.43% accuracy in detecting cancer in histopathology pictures by implementing deep learning using a lightweight MicroMobileNet on a mobile device. The new network was installed on an edge device with a fast speed (140 ms) and very little memory (7.4 MB).

3. METHODOLOGY

3.1 Dataset preprocessing

In our work, we collected the chest X-ray images dataset from National Institute healthcare NIH chest X-ray dataset website. The NIH Clinical Center contributed the data, which can be downloaded from the NIH website at <https://nihcc.app.box.com/v/ChestX-rayNIHCC>, the images with PNG file format, the images were pre-processed by resizing the images from the size 1024×1024×3 to (224×224×3) dimension and then normalized. For normalization, the pixel values were scaled to the [0,1] range, this was achieved by dividing each pixel by 255. This ensures compatibility with MobileNetV2 models and increases convergence stability. We also augmented the training split of dataset to prevent overfitting. Four operations were implemented to expand the training set of 1626 photos (70% of the original 2323-image dataset): up-down side flipping, repeated rotations (90°, 180°, 270°) following flipping, a unique black-and-white conversion, and consecutive rotations (90°, 180°, 270°). To improve feature extraction for the model, the converter employed a computed pixel threshold of $((\max - \min)/2) + \min$, when max and min represented the maximum and minimum values of pixels respectively, designating pixels above the threshold as white (255) and those below as black (0). In Figure 2, we show samples of NIH dataset used in proposed method that were taken from NIH dataset.

We collected 2323 images and selected them among 8000 chest X-ray images samples, we collected these samples from NIH for training MobileNetV2 model, we labeled these input images dataset as two groups, first group included (normal or not finding) images, the second group of the input images dataset was labeled as (mass) images that include (mass, nodule, tumor, fibrosis). We trained both models of MobileNetV2 (pretrained and scratch trained) by these input image dataset groups. The target is to classify chest X-ray

images into two classes: normal and mass, the classification was done according to input group of images dataset, the dataset was splitted into 3 sets, training set, validation set and testing set. 70% of dataset used for training, 15% used for validation and 15% used for testing, we trained each model and validated at 10 epochs with batch size of 32, we trained both models of MobileNetV2 and deployed them on Raspberry Pi 5, we used it as limited constrained resource hardware. We implemented our work by Python software version 3.7, we imported the Pytorch software library and parameter server of deep learning that support Raspberry Pi platform. The methodology of proposed approach can be shown in Figure 3.

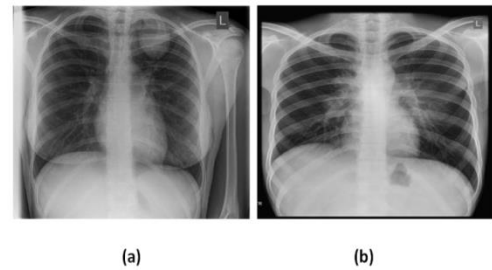


Figure 2. Samples of NIH X-ray images dataset: a) lung mass; b) normal lung

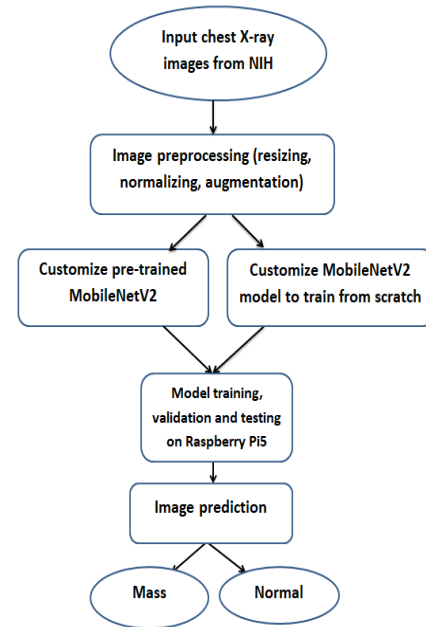


Figure 3. Proposed method of tiny deep learning

3.2 MobileNetV2 model

In our method, we implemented the MobileNetV2 [26] and trained from scratch and also we used a pretrained MobileNetV2, we trained both models by the same dataset and then deployed them on Raspberry Pi 5, MobileNetV2 is introduced as enhanced version of MobileNetV1 that makes it more powerful and efficient. In architecture of the MobileNetV2, the block of the depthwise separable convolution has been re-designed as shown in Figure 4. In the new depthwise separable convolution block there are 3 convolutional layers. Prior to entering the depthwise

convolution layer, the input feature map's channel count is increased in the first layer, a 1×1 convolution layer. The intermediate layer, a 3×3 depthwise convolution layer similar to the MobileNetV1, filters the input feature map. However, the input feature map's channel count will be decreased because the data is projected with a large number of channels into a tensor with a significantly smaller number of channels in the final layer, which is made up of a 1×1 convolution layer. Because less data will be moving across the network, this last layer is also known as a bottleneck layer. In addition, to help with the flow of gradients through the network, the residual connection as in ResNet [27] is adopted in the MobileNet v2 architecture. In the new depthwise separable convolution, each layer is followed by batch normalization and ReLU6 as activation function (using non linearity after the last bottleneck will lose useful information, hence it cannot be used in this layer). The 17 new depthwise separable convolution blocks in MobileNet v2 are followed by a standard 1×1 convolution layer. A standard 3×3 convolution layer with 32 channels makes up the first layer.

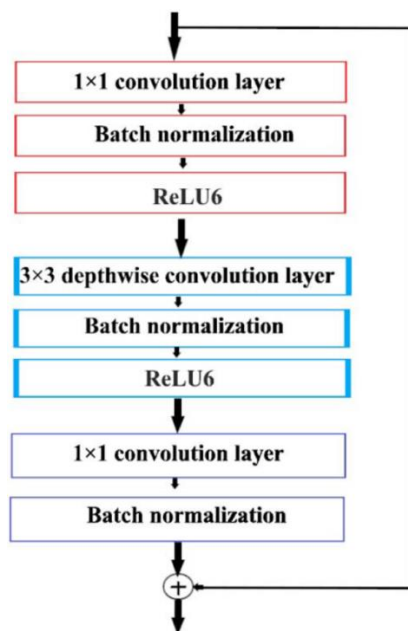


Figure 4. Fundamental functional unit of MobileNetV2 [26]

3.3 Hardware requirement-Raspberry Pi 5

In our work, we used the MobileNetV2 model to detect lung mass, we deployed MobileNetV2 on latest version of Raspberry Pi (Raspberry Pi 5), we used Raspberry Pi 5 as limited constrained resource. Raspberry Pi 5 was protected by cover with active cooler of 2 fans to cool the system, Raspberry Pi 5 draws 5 Ampere (Amp) as maximum current, so, the power supply should be capable of drawing that much instantaneously. The power needed is ($5 \text{ Amp} * 5 \text{ Volt(V)} = 25$) Watt. Also the power supply should be uninterrupted while it operates, which it means the power supply should be continuous and must be provided using batteries, and because of instability of electric power here in Iraq, at the first, we connected power bank of 10 Amp/hour as a power supply of our Raspberry Pi 5, but this power bank couldn't provide adequate time for the training process to train MobileNetV2 from scratch and it enabled the Raspberry Pi 5 to operate 8 hours only, so, we designed a step down power supply of dc-dc converter as shown in Figure 5 to supply power for

Raspberry Pi 5 instead of the power bank, this dc-dc converter power supply consist of 4 stack sealed acid batteries we connected these batteries in parallel, each battery delivers 12V and 5 (Amp/h) of total power equal to 240 watt, we finally connected these batteries to voltage regulator module XL4015 and we set this regulator to deliver 5 V for Raspberry Pi 5, thus, this step down converter power supply will deliver 48 Amp/h and it was enough for training time process that was required for each epoch, each epoch required approximately more than 3 hours to train MobileNetV2 from scratch and more than 2 hours for pretrained MobileNetV2 as shown in Table 1 that referred to the training time consumed by both models in hours, minutes and seconds ("h:min:s").

Table 1. The training time consumed by both scratch trained and pretrained MobileNetV2 models for 10 epochs in our proposed study

No. of Epochs	Training Time Consumed by Trained Scratch MobileNetV2 Model Displayed as "h:min:s"	Training Time Consumed by Pretrained MobileNetV2 Model Displayed as "h:min:s"
1	24:02:06	4:46:12
2	3:04:12	2:33:18
3	3:00:18	2:33:18
4	3:14:42	2:32:42
5	3:11:06	2:30:00
6	3:04:30	2:30:18
7	3:12:18	2:29:42
8	3:11:06	2:29:42
9	3:06:54	2:33:00
10	3:03:54	2:29:42

It can be shown from Table 1 that the training time consumed by pretrained model is less than the training time taken by scratch trained model.

We added an Additional secure digital SD-card of 64 GB to Raspberry Pi 5 to get more memory storage for Operation System OP, dataset, model, code, results...etc, and we installed Debian Linux Operating System OP in Raspberry Pi 5, in the beginning, we connected Raspberry Pi 5 to desktop monitor through Micro-HDMI cable to complete setting and software installation because there is no HDMI port in our laptop personal computer PC, and then, we connected Raspberry Pi 5 wirelessly to our laptop PC and we used as portable monitor and control panel for Raspberry Pi 5, we made the method of wireless connection by connecting Raspberry Pi 5 to the monitor of our PC through making our mobile phone as a router by activating the communication and conduction point in the mobile phone device and then logging through the internet network account that have been identified previously on desktop monitor for Raspberry Pi 5, we identified the Internet Protocol IP of Raspberry Pi 5 in PC by using IP scanner software application in PC, we inserted this IP into Real Virtual Network Connection VNC software in PC to complete wireless connection method. We implemented all our proposed method by python version 3.7 and Visual Studio Code (VSC), we used it as updated integrated developed environment IDE for Python, Thonny is, a traditional Python IDE for Raspberry Pi intended for novices, it was substituted with VSC IDE. VSC was designed specifically for the Python programming language and is written in Python, to assist users in writing and testing Python code, VSC offers an intuitive interface along with a number of features like syntax highlighting, code completion, and debugging tools. The

specification and features of Raspberry pi 5 hardware of our proposed study is shown in Table 2.

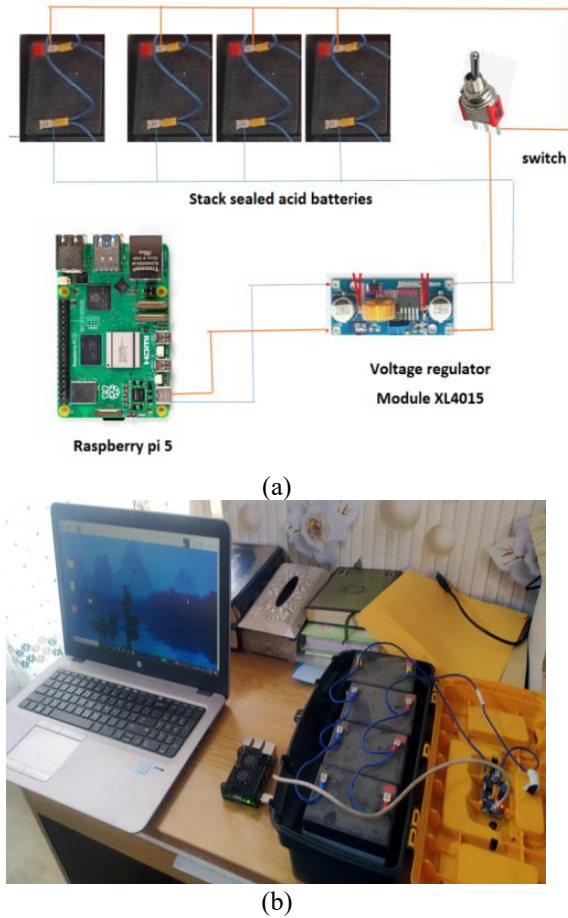


Figure 5. (a) Design of Raspberry Pi 5 power supply; (b) Proposed hardware system

Table 2. The specification of Raspberry Pi 5 that used in our proposed study

Raspberry Pi 5 Features
A 5v/5A USB-C power source is advised; a 5v/3A minimum is needed.
2 micro HDMI ports supports up to 4kp60
LPDDR4X SD RAM 8 GB
CPU Quad Arm Cortex-A76, 2.4 GHz
SWAP file of 4 GB
No GPU
2 USB 3.0 ports
2 USB 2.0 ports
Gigabit Ethernet port
PoE-capable 802 11b/g/n/ac wireless requires a poE HAT, which is supplied separately.
PCIe expansion connector requires PCIe adaptor, sold separately
64-bit quad core cortex-A76processor

4. RESULTS AND DISCUSSION

As mentioned in methodology, we trained and deployed the two models: trained from scratch MobileNtev2 and pretrained MobileNetV2 on Raspberry Pi 5 that we used as limited constrained resource hardware. The performance parameters was calculated according to equations of accuracy, precision, Recall, F1 score shown in Eqs. (1)-(4) below.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{F1Score} = \frac{2 * \text{precision} * \text{Recall}}{\text{precision} + \text{Recall}} \quad (3)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4)$$

When accuracy represents the summation of true positive (TP) and true negative values (TN) divided by the total components of the confusion matrix (TP, TN, false positive FP and false negative FN). Precision is a ratio of total number of cases that expected to be positive and the number of cases that are really positive. Recall is called also the sensitivity; it can be defined as the portion of the total number of cases of correctly classified positive to the total number of positive cases. F1 score is can be defined as the proportional average of accuracy and Recall, training and validation behavior of scratch trained MobileNetV2 can be shown in Figure 6 and Figure 7, respectively.

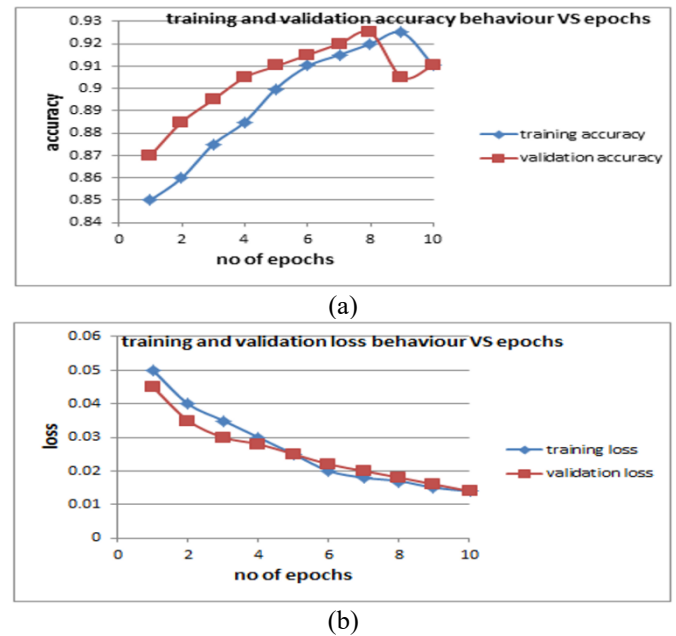


Figure 6. (a) Training and validation accuracy behavior VS 10 epochs of MobileNetV2 trained from scratch; (b) Training and validation loss VS 10 epochs of MobileNetV2 trained from scratch on Raspberry Pi 5

Figure 6(a) makes it evident that the training and validation accuracy rises with the number of epochs. This behavior is usually observed when the model starts to learn features and patterns from training image samples. It is also possible to see that the validation accuracy is marginally higher than the training accuracy, which suggests that MobileNetV2 has good generalization to built-in regularization techniques like batch normalization and depth-wise separable convolutions. It can be seen that both the validation and training accuracy reach more than 91%. Figure 6(b) shows as the number of epochs rose, the training and validation loss reduced as well, reaching 0.015 approximately. This suggests that the MibileNetv2 is learning effectively.

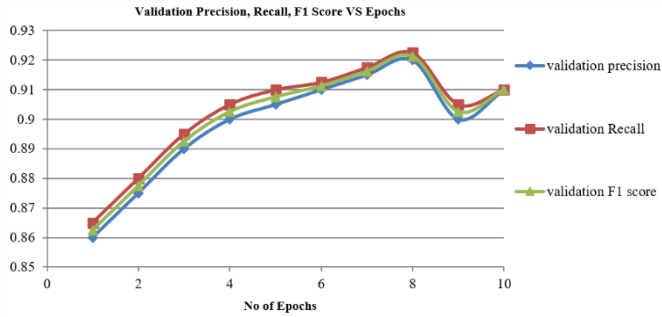


Figure 7. Validation precision, Recall and F1 score behavior VS 10 epochs of MobileNetV2 trained from scratch on Raspberry Pi 5

It is clearly that from Figure 7, the performance parameters (precision, Recall, F1 score) increased when number of epochs increased and goes more than 91%, this indicate to good performance of MobileNetV2 that was trained from scratch, the performance confusion matrix of test dataset of MobileNetV2 trained from scratch can be shown in Figure 8.

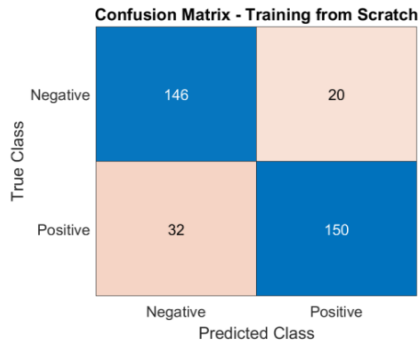


Figure 8. Confusion matrix of test dataset of MobileNetV2 trained from scratch

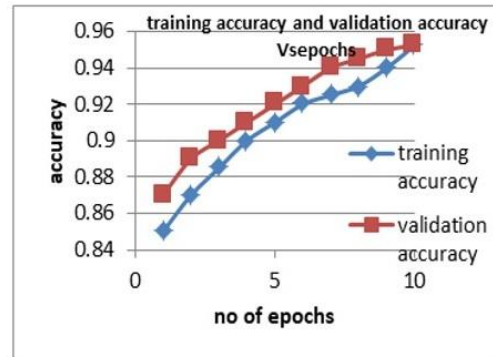
From Figure 8, it can be shown that the parameters values of confusion matrix of test dataset are:

True Positives (TP):150, True Negatives (TN): 146, False Positives (FP): 20, False Negatives (FN): 32. This distribution sums to $150+146+20+32=348$ sample (represent the test data set 15% of total samples). So, Accuracy: $(TP+TN)/Total=(150+146)/348=296/348\approx0.8506$ or 85.06%. Precision: $TP/(TP+FP)=150/(150+20)=150/170\approx0.8824$ or 88.24%. Recall (Sensitivity): $TP/(TP+FN)=150/(150+32)=150/182\approx0.8242$ or 82.42%. F1-Score: $2*(Recall*Precision)/(Recall+Precision)=2*(0.8242*0.8824)/(0.8242+0.8824)\approx2*0.7272/1.7066\approx0.8521$ or 85.21%.

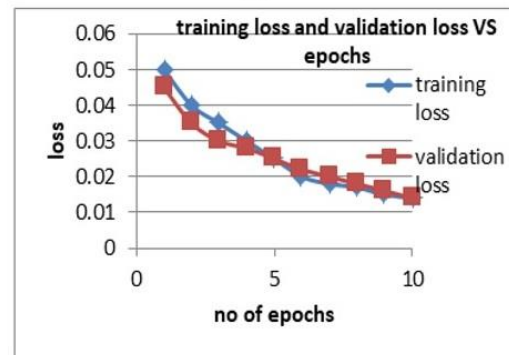
The training and validation behavior of pretrained MobileNetV2 can be shown in Figure 9 and Figure 10 respectively.

It is obviously from Figure 9(a) that the accuracy of validation is slightly higher than training accuracy and this indicate to good generalization of pretrained MobileNetV2 on Raspberry Pi 5 and belong to regularization effect of this model due to technique of depth wise convolutions and batch normalization technique, it is clearly that validation accuracy reached to more than 95% and it's higher than the validation accuracy of MobileNetV2 trained from scratch, this indicate that the model perform better and learn faster because the early layers which has been already adapted to general feature extraction, while scratch trained MobileNetV2 model start with random weights and learns feature extraction directly

from lung mass dataset, so, the scratch trained model will face difficulty to learn complex patterns with this limited dataset, also from Figure 9(b), it's clearly that training and validation loss reached to 0.015 approximately and it is the same of trained scratch model because both models are optimizing the loss function effectively, this confirm to well learning of this model in Figure 10, it can be shown the validation precision, Recall and F1-score behavior in 10 epochs.



(a)



(b)

Figure 9. (a) The accuracy behavior of training and validation of pretrained MobileNetV2 for 10 epochs; (b) Training and validation loss of pretrained MobileNetV2 for 10 epochs

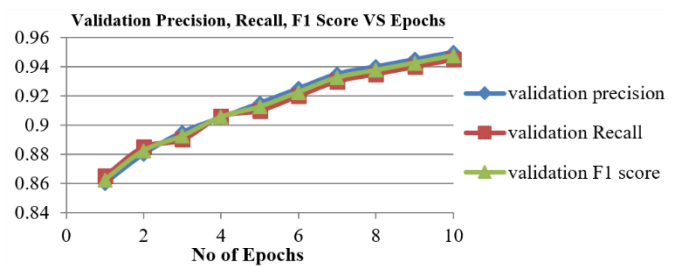


Figure 10. Validation precision, Recall and F1-score behavior VS 10 epochs of pretrained MobileNetV2 on Raspberry Pi 5

It is clearly that from Figure 10, the performance parameters (precision, Recall, F1 score) increased when number of epochs increased and goes more than 94% and this is higher than performance parameters value of trained scratch model, this is also indicate to good performance of MobileNetV2 that was trained from scratch, Figure 11 show the confusion matrix of test dataset performance of pretrained MobileNetV2.

From Figure 11, it's clearly that the parameter's values of confusion matrix of test dataset of pretrained MobileNetV2

are: True Positives (TP): 160, True Negatives (TN): 153, False Positives (FP): 15, False Negatives (FN): 20. This distribution sums to 160+153+15+20=348 samples (represent the test data set 15% of total samples), so, Accuracy: $(TP+TN)/Total=(160+153)/348=313/348\approx 0.8994$ or 89.94%, Precision: $TP/(TP+FP)=160/(160+15)=160/175\approx 0.9143$ or 91.43%, Recall or (Sensitivity): $TP/(TP+FN)=160/(160+20)=160/180\approx 0.8889$ or 88.89%, F1-Score: $2*(Recall*Precision)/(Recall+Precision)=2*(0.9143*0.8889)/(0.9143+0.8889)\approx 2*0.8127/1.8032\approx 0.9014$ or 90.14%.

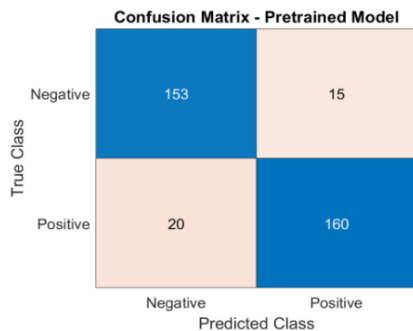


Figure 11. Confusion matrix of test dataset of pretrained MobileNetV2

It’s clearly that the pretrained MobileNetV2 performed better on test data than the scratch trained model, the pretrained model not only generalized better but also achieved a clinically more acceptable balance between false positives and false negatives, the reduction in false negatives from 32 in scratch model to 20 in the pretrained model is particularly significant, as missed detection of lung masses could lead to late diagnosis and treatment.

Pretrained models make use of transfer learning, in which weights from extensive datasets (like ImageNet) are used to initiate convolutional layers. Rich and generalized low- and mid-level features, like edges, textures, and forms, are already captured by these pretrained layers and can be applied to medical images. Compared to training from scratch, when the network must learn all features from random initialization, this background information enables the model to converge more quickly, use less data, and achieve higher accuracy.

4.1 Computation and energy load analysis

4.1.1 MAC operations

- MobileNetV2 performs approximately 300 million inference multiply-accumulate operations (MACs) per $224\times224\times3$ image.
- For each epoch, training step handles approximately 13.17 trillion MACs. This is calculated by (multiplying 300 million by 3 (for training MACs that included forward and backward pass which equal to 3 times of inference MACs) and then multiply the result by 14634 images (which represent 70% of training dataset 1626 images multiplied by 9), when 9 represents the 8 operations of augmentation (twice rotation and one flipping and one converted operation) plus the original training dataset.

4.1.2 Power and energy estimation

- Raspberry Pi 5 consumed up to 6.4 Watt under full CPU load, so, the measured value of current during full CPU load was around 1.28 Amp.

- Energy usage per epoch was estimated by multiplying the power drawn from device (6.4 Watt) by the training time per epoch (in hours) as shown in Table 3 that summarizes the estimated MACs and power consumption across 10 training epochs.

Table 3. Power and energy consumption of pretrained MobileNetV2 across 10 epochs

Epoch	Max Power for CPU Load in Watt	Time (h)	Energy (Watt Hour)	MACs/Epoch (Trillions)
1	6.4	4.770	30.53	13.17
2	6.4	2.555	16.35	13.17
3	6.4	2.555	16.35	13.17
4	6.4	2.545	16.29	13.17
5	6.4	2.500	16.00	13.17
6	6.4	2.505	16.03	13.17
7	6.4	2.495	15.97	13.17
8	6.4	2.495	15.97	13.17
9	6.4	2.550	16.32	13.17
10	6.4	2.495	15.97	13.17

It’s clearly that from Table 3, the total number of MACs of 10 epochs was 131.7 trillion MAC, also, the total energy that consumed in 10 epochs was approximately 175.78 Watt hour (Wh), each Watt hour equivalent to 3600 Joule, so, the total energy in Joule= 632808 Joule, then the energy consumed per MAC can be obtained from division of 632808 Joule by 131.7 trillion MACs, so the energy consumed per MAC will be 4.803 nano Joule/MAC.

Furthermore, the computational cost of Raspberry Pi 5 is still lower, so, over ten epochs, the Raspberry Pi 5 consumed only 6.4 Watt and 175.78 Watt hour (about 0.176 kilo Watt h (kWh)) to train model. Contrast this with: super computers (that consume kilo Watts and megawatts power), or large model training on the NVIDIA V100 or A100 that can quickly consume more power (about 350-500 Watt), and also consumed more or the same energy in (Watt hours) for the same small task in our work (when a small dataset and small model were used) that makes these traditional hardware to be overkill for small tasks, this clearly support power and energy efficient and low computational cost deep learning in embedded systems.

Moreover, the Raspberry Pi 5 offers low cost and requires minimal infrastructure, with its price around \$100, whereas the price of an NVIDIA A100 server is about \$15,000, not including additional costs such as cooling systems and data center power.

So, the Raspberry Pi 5 is appropriate for low-resource, edge-based, and sustainable deployments because it provides competitive AI performance for medical imaging workloads utilizing a fraction of the overall energy and hardware cost. When compared to conventional deep learning systems that depend on GPUs or supercomputers, the training done on a Raspberry Pi 5 demonstrates a noticeable decrease in energy and operating costs. The Raspberry Pi offers a very energy-efficient substitute for training or optimizing lightweight models in limited settings, even though it cannot match the sheer speed or capacity of such systems. This finding bolsters the expanding sustainable AI movement and highlights how low-power edge devices might increase deep learning's affordability, accessibility, and environmental friendliness particularly for medical applications in distant or underdeveloped locations. To further contextualize the measured energy efficiency of our proposed Raspberry Pi 5-

based MobileNetV2 system, we compared its energy per MAC and MACs/Watt with a selection of edge AI platforms and model deployments reported in the literature Table 4. Our pretrained MobileNetV2 achieves 4.803 nJ/MAC, equivalent to $\sim 2.08 \times 10^8$ MACs/Watt, during on-device training and inference. This performance is competitive with many embedded AI accelerators while maintaining the advantages of general-purpose programmability. For example, Jetson Nano running ResNet-18 achieves ~ 93 nJ/MAC ($\sim 1.07 \times 10^9$ MACs/Watt) [28] and Google Edge TPU running MobileNetV2 achieves ~ 73 nJ/MAC ($\sim 1.36 \times 10^9$ MACs/Watt), but these accelerators are limited to inference-

only workloads. Raspberry Pi 4 running MobileNetV1 demonstrates lower efficiency (~ 150 nJ/MAC; $\sim 6.67 \times 10^8$ MACs/Watt) [29].

These results show that although certain specialized accelerators can achieve higher absolute MACs/Watt, the proposed Raspberry Pi 5 system offers a favorable trade-off between energy efficiency, hardware accessibility, and full training capability—qualities critical for real-world deployment in rural or resource-constrained medical environments. This balance supports sustainable AI solutions while avoiding the infrastructure requirements of GPU servers or FPGA tool chains.

Table 4. The comparison with other studies

Study	Model Used	Dataset	Deployment Platform	Accuracy	Performance Metrics	Study Finding
This study	MobileNetV2 (pretrained & scratch)	NIH Chest X-ray dataset (2322 images, binary: mass vs. normal)	Raspberry Pi 5 (64-bit quad-core Cortex-A76, 8 GB RAM)	89.94% (pretrained) 85.06% (scratch)	Precision: 91.43% F1-Score: 90.14% Recall: 88.89% 4.803 nJ/MAC ($\sim 2.08 \times 10^8$ MACs/Watt) (pretrained)	On-device training & inference; sustainable edge deployment.
Hadidi et al. [28]	ResNet-18, MobileNetV2	ImageNet	Jetson Nano, Google Edge TPU	69.8%, 71.8%	93 nJ/MAC, 73 nJ/MAC	Edge inference baseline; moderate efficiency.
Dunkel et al. [30]	ResNet-50	ImageNet	Jetson Nano (GPU)	76.2%	Est. 120 nJ/MAC ($\sim 8.33 \times 10^8$ MACs/Watt)	Better than CPU-only inference, but less efficient than TPU/FPGA.
Mohammed [19]	MobileNet + CNN + Energy Valley Optimization	NIH Chest X-ray dataset	Standard PC with TensorFlow backend	85.91%	Precision: Not available Recall: Not available, No energy/power metric specified	Introduced metaheuristic optimization; no actual deployment or edge feasibility tested.
Soudi et al. [21]	Modified MobileNetV2	NIH Chest X-rays (multi-label)	Google Colab GPU (Tesla K80)	90%	AUC: 0.811 F1-score: not available, No energy/power metric specified	Improved performance on NIH data; no hardware constraint, no power profiling.
Baller et al. [29]	MobileNetV1	ImageNet	Raspberry Pi 4	70.6%	Est. 150 nJ/MAC ($\sim 6.67 \times 10^8$ MACs/Watt)	General-purpose board; less efficient than AI accelerators.
Sandler et al. [31]	MobileNetV2	ImageNet	Smartphone (Pixel 4)	71.8%	Est. 90 nJ/MAC ($\sim 1.11 \times 10^9$ MACs/Watt)	Mobile deployment feasible; efficiency depends on SoC NPU.
Kumar et al. [24]	MobiHisNet (custom lightweight CNN)	Histopathological images (breast cancer)	Raspberry Pi 2 (ARM Cortex-A7)	Reported "Good accuracy" (value not specified)	Low memory & cost-efficient histopathology deployment, No energy/power metric specified	Validated DL on Pi2, but older device & unrelated to lung imaging.
Tan and Le [10]	EfficientNet-B0	ImageNet	NVIDIA Jetson Nano	77.1%	Est. 110 nJ/MAC ($\sim 9.09 \times 10^8$ MACs/Watt)	Better accuracy but slightly lower efficiency than MobileNet.
Hou and Navarro-Cia [32]	EfficientNetB0	CT images from COVID19-CT dataset to detect COVID19	Intel(R) Core i7-10875H, NVIDIA GeForce RTX 2060 6G	91.15%-95.5%	AUC of 96.4%-98.5% No energy/power metric specified	Improved classification performance, no limited hardware resource.
Wang et al. [33]	SqueezeNet	Mammography images from MIAS dataset for breast cancer	Workstation with GPU	94.1%	Sensitivity of 94.3%, no energy metric specified	Hybrid model for effectively diagnostic, no constraint device validation

Generalizability may be limited by the very small dataset (2323 photos) used in this study, which came from a single public source. Inference speed and robustness in real-world

scenarios are yet unknown because real-time testing and clinical integration were not assessed. Future research will include clinical validation in partnership with healthcare

organizations, real-time deployment on Raspberry Pi 5, and bigger, more varied datasets.

5. CONCLUSIONS

The Raspberry Pi 5's capacity to support lightweight CNN models for medical image processing was demonstrated by the successful training and testing of both models on the device. This supports its application in actual offline or low-resource healthcare settings, particularly those with restricted access to cloud computing. It can be concluded that the achievement of pretrained MobileNetV2 model is better than MobileNetV2 trained from scratch, this is demonstrating the advantage of transfer learning of the pretrained MobileNetV2, the prior knowledge of pretrained MobileNetV2 help the model to generalize better even with relatively limited dataset and allows the model to extract more meaningful features with limited resource, so, the pretrained MobileNetV2 is more comfortable for Raspberry Pi 5 platform to make efficient portable tiny deep learning system to predict lung mass disease and enable low expert doctors who working in the rural medical centers for lung tumors diagnosis using low computational cost hardware and low cost chest X-ray exam with a competitive obtained accuracy value if it was compared with other Artificial intelligent models that require a high level hardware and high computational cost. Furthermore, the training pipeline's successful implementation on the Raspberry Pi 5 shows that low-cost, portable AI systems for medical diagnostics are feasible. These findings support the larger objective of democratizing medical AI and advance the expanding field of AI-powered point-of-care imaging.

For future research attempting to integrate deep learning, transfer learning, and embedded systems for practical healthcare applications, the study thus offers a useful resource.

In future work, accuracy may be enhanced when pretrained MobileNetV2 is fine-tuned and performance of both models can be enhanced by increasing the size of training dataset and by deploying other light weight CNN models.

AUTHOR CONTRIBUTIONS

Conceptualization, methodology, software, and validation, Yasir Salam Abdulghafoor; formal analysis, investigation, resources, data curation, Yasir Salam Abdulghafoor; and Ahmed Faeq Hussein; writing—original draft preparation, writing—review and editing, visualization, Yasir Salam Abdulghafoor; supervision, Auns Qusai Al-Neami, project administration, Auns Qusai Al-Neami and Ahmed Faeq Hussein; funding acquisition, Yasir Salam Abdulghafoor.

REFERENCES

- [1] Jan, A., Ahmed, N., Awan, N.I., Khan, B., Ul-Islam, M., Ahmed, I., Shabbir, A., Mohammad, A., Shah, H. (2021). The lung mass and nodule: A case series: The lung mass and nodule. *Pakistan BioMedical Journal*, 4(1): 37-43. <https://doi.org/10.52229/pbmj.v4i1.61>
- [2] Vinushree, S., Gowda, R.M. (2019). Segmentation of lung cancer using deep learning. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(2): 1188-1192. <https://doi.org/10.35940/ijrte.B1849.078219>
- [3] Alakwaa, W., Nassef, M., Badr, A. (2017). Lung cancer detection and classification with 3D convolutional neural

- network (3D-CNN). *International Journal of Advanced Computer Science and Applications*, 8(8): 409-417. <https://doi.org/10.14569/IJACSA.2017.080853>
- [4] Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, 9(4): 611-629. <https://doi.org/10.1007/s13244-018-0639-9>
- [5] Xiao, P., Qin, Z., Chen, D., Zhang, N., Ding, Y., Deng, F., Pang, M. (2023). FastNet: A lightweight convolutional neural network for tumors fast identification in mobile-computer-assisted devices. *IEEE Internet of Things Journal*, 10(11): 9878-9891. <https://doi.org/10.1109/IJOT.2023.3235651>
- [6] Han, H., Siebert, J. (2022). TinyML: A systematic review and synthesis of existing research. In *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIC)*, Jeju Island, Korea, pp. 269-274. <https://doi.org/10.1109/ICAIC54071.2022.9722581>
- [7] Lê, M.T., Wolinski, P., Arbel, J. (2023). Efficient neural networks for tiny machine learning: A comprehensive review. *arXiv preprint arXiv:2311.11883*. <https://doi.org/10.48550/arXiv.2311.11883>
- [8] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv preprint arXiv:1602.07360*. <https://doi.org/10.48550/arXiv.1602.07360>
- [9] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. <https://doi.org/10.48550/arXiv.1704.04861>
- [10] Tan, M., Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, PMLR, pp. 6105-6114.
- [11] Abedin, A., Ehsanpour, M., Shi, Q., Rezatofighi, H., Ranasinghe, D.C. (2021). Attend and discriminate: Beyond the state-of-the-art for human activity recognition using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1): 1-22. <https://doi.org/10.1145/3448083>
- [12] Georgiev, P., Bhattacharya, S., Lane, N.D., Mascolo, C. (2017). Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3): 1-19. <https://doi.org/10.1145/3131895>
- [13] Liu, Y., Lin, C., Li, Z. (2021). WR-hand: Wearable armband can track user's hand. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3): 1-27. <https://doi.org/10.1145/3478112>
- [14] Yin, H., Zhou, A., Su, G., Chen, B., Liu, L., Ma, H. (2020). Learning to recognize handwriting input with acoustic features. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2): 1-26. <https://doi.org/10.1145/3397334>
- [15] Chowdhery, A., Warden, P., Shlens, J., Howard, A., Rhodes, R. (2019). Visual wake words dataset. *arXiv*

- preprint arXiv:1906.05721.
<https://doi.org/10.48550/arXiv.1906.05721>
- [16] Liu, X., Li, Y., Fromm, J., Wang, Y., Jiang, Z., Mariakakis, A., Patel, S. (2021). SplitSR: An end-to-end approach to super-resolution on mobile devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1): 1-20. <https://doi.org/10.1145/3448104>
 - [17] Ghosh, R., Ahamed, A., Sadhukhan, B., Das, N. (2023). Lung nodule classification using mobilenet transfer learning. In *2023 9th International Conference on Smart Computing and Communications (ICSCC)*, Kochi, Kerala, India, pp. 290-295. <https://doi.org/10.1109/ICSCC59169.2023.10335043>
 - [18] Mothkur, R., Veerappa, B.N. (2023). Classification of lung cancer using lightweight deep neural networks. *Procedia Computer Science*, 218: 1869-1877. <https://doi.org/10.1016/j.procs.2023.01.164>
 - [19] Mohammed, O.N. (2024). Enhancing pulmonary disease classification in diseases: A comparative study of CNN and optimized MobileNet architectures. *Journal of Robotics and Control (JRC)*, 5(2): 427-440. <https://doi.org/10.18196/jrc.v5i2.21422>
 - [20] Rahman, M.S., Shill, P.C., Homayra, Z. (2019). A new method for lung nodule detection using deep neural networks for CT images. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Cox'sBazar, Bangladesh, pp. 1-6. <https://doi.org/10.1109/ECACE.2019.8679439>
 - [21] Souid, A., Sakli, N., Sakli, H. (2021). Classification and predictions of lung diseases from chest X-rays using mobilenet V2. *Applied Sciences*, 11(6): 2751. <https://doi.org/10.3390/app11062751>
 - [22] Mengistie, T.T., Kumar, D. (2021). Comparative study of transfer learning techniques for lung disease prediction. In *2021 10th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON)*, Jaipur, India, pp. 1-6. <https://doi.org/10.1109/IEMECON53809.2021.9689159>
 - [23] Yaqoob, M., Qayoom, H., Hassan, F. (2021). Covid-19 detection based on the fine-tuned MobileNetv2 through lung X-rays. In *2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, Alkhobar, Saudi Arabia, pp. 1-6. <https://doi.org/10.1109/ISAECT53699.2021.9668425>
 - [24] Kumar, A., Sharma, A., Bharti, V., Singh, A.K., Singh, S.K., Saxena, S. (2021). MobiHisNet: A lightweight CNN in mobile edge computing for histopathological image classification. *IEEE Internet of Things Journal*, 8(24): 17778-17789. <https://doi.org/10.1109/JIOT.2021.3119520>
 - [25] Biswas, S., Barma, S. (2023). MicrosMobiNet: A deep lightweight network with hierarchical feature fusion scheme for microscopy image analysis in mobile-edge computing. *IEEE Internet of Things Journal*, 11(5): 8288-8298. <https://doi.org/10.1109/JIOT.2023.3317878>
 - [26] Nguyen, H. (2020). Fast object detection framework based on mobilenetv2 architecture and enhanced feature pyramid. *Journal of Theoretical and Applied Information Technology*, 98(5): 812-824.
 - [27] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 770-778. <https://doi.org/10.1109/CVPR.2016.90>
 - [28] Hadidi, R., Cao, J., Xie, Y., Asgari, B., Krishna, T., Kim, H. (2019). Characterizing the deployment of deep neural networks on commercial edge devices. In *2019 IEEE International Symposium on Workload Characterization (IISWC)*, Orlando, FL, USA, pp. 35-48. <https://doi.org/10.1109/IISWC47752.2019.9041955>
 - [29] Baller, S.P., Jindal, A., Chadha, M., Gerndt, M. (2021). DeepEdgeBench: Benchmarking deep neural networks on edge devices. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*, San Francisco, CA, USA, pp. 20-30. <https://doi.org/10.1109/IC2E52221.2021.00016>
 - [30] Dunkel, E.R., Swope, J., Candela, A., West, L., Chien, S.A., Towfic, Z., Fernandez, M.R. (2023). Benchmarking deep learning models on myriad and snapdragon processors for space applications. *Journal of Aerospace Information Systems*, 20(10): 660-674. <https://doi.org/10.2514/1.1011216>
 - [31] 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*, Salt Lake City, UT, USA, pp. 4510-4520. <https://doi.org/10.1109/CVPR.2018.00474>
 - [32] Hou, Y., Navarro-Cia, M. (2023). A computationally-inexpensive strategy in CT image data augmentation for robust deep learning classification in the early stages of an outbreak. *Biomedical Physics & Engineering Express*, 9(5): 055003. <https://doi.org/10.1088/2057-1976/ace4cf>
 - [33] Wang, J., Khan, M.A., Wang, S., Zhang, Y. (2023). SNSVM: SqueezeNet-guided SVM for breast cancer diagnosis. *Computers, Materials & Continua*, 76(2): 2201-2216. <https://doi.org/10.32604/cmc.2023.041191>