NOTATION TRENDPLUS PUBLISHING
WHERE IDEAS MEET IMPACT

Volume-1, Issue 2, August 2026, Page No- 94-112
Journal of Machine Learning Innovations and Artificial Intelligence Horizons (JMLIAIH)

Lightweight Edge AI Framework for Real-Time Disease Prediction on Resource-Constrained Healthcare Devices

Dr. Amitava Biswas

Department of Computer Science, Vivekananda College Thakurpukur, 269, Diamond Harbour Road, Thakurpukur, Kolkata, 700063, West Bengal.

DOI: https://doi.org/10.70903/jmliaih.2026.1206

Article Received: 11-07-2026
Revised Version: 18-08-2026
Accepted: 25-08-2026

*Corresponding Author

Dr. Amitava Biswas

E-Mail Id:
abiswas.seminar@gmail.com

Copyright: © The Author(s), 2026. Published by Notation Trendplus Publishing Pvt. Ltd. This is an Open Access article, distributed under the terms of the Creative Commons Attribution 4.0 License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

Abstract: The conflicting requirements of high diagnostic accuracy and limited computational resources pose a big challenge in real-time detection of disease, especially for resource constrained healthcare devices. We suggest a thin-edge AI solution that allows an efficient prediction of the disease in a certain device located at the edge of the network without depending on cloud connectivity. The framework starts with a medical images and numerical clinical data tailored pre-processing pipeline which consist of normalization, augmentation and noise reduction. In essence, we use a small deep learning model like MobileNetV3 or EfficientNet-Lite that use depthwise separable convolutions that reduce a model's floating point operations and parameters by many orders of magnitude. We then implement a multi-stage compression strategy to further take into account the constraints of the edge devices. Post-training quantization converts weights and activations from 32-bit to 8-bit integer representation, while structured pruning removes redundant connections via a binary mask. Knowledge distillation then transfers representational knowledge from a larger teacher model to the compact student model, optimizing a combined loss of cross-entropy and Kullback-Leibler divergence. The final compressed model is deployed on platforms like the NVIDIA Jetson Nano or ARM-based microcontrollers, where inference executes locally using a sigmoid or softmax activation for binary or multi-class classification, respectively. The primary contribution of this work is a holistic integration of architectural efficiency, compression techniques, and edge deployment that maintains diagnostic accuracy while achieving minimal latency and zero network dependency. Experimental results demonstrate that the proposed framework significantly reduces model size and inference time compared to standard approaches, making it suitable for real-time applications in remote or low-resource clinical settings. This work therefore provides a practical pathway toward democratizing AI-driven healthcare diagnostics

Keywords: Edge AI, Deep Learning, Real-Time Disease Prediction, Resource-Constrained Devices, Healthcare Applications.

Introduction

The rise of artificial intelligence has been undoubtedly a great promise for improving the accuracy and effectiveness in diagnosis in the health sector, particularly in medical imaging and physiological data (Shen et al., 2017). Some of the traditional deep learning approaches, such as the Convolutional Neural Networks (CNNs) or the Recurrent Neural Networks (RNNs), have been applied to the most advanced level of art in the field of imaging to disease progression prediction (Che et al., 2018), to mention a few. These models usually have very large numbers of millions of parameters and many times may require vast amounts of computational resources, that is why often large cloud-based inference is required. The dependence of these services in cloud connectivity causes some issues in the resource-limited healthcare settings, such as latency, bandwidth constraints and critical issues related to privacy and security of patients' data (Rahmani et al., 2018).

To overcome these drawbacks, ‘edge computing' paradigm has become an attractive approach for data processing and data inference to move close to the source—in microcontroller, embedded systems, or even a single-board computers. Onboard AI models on edge devices provide benefits such as real-time responsiveness, reduced network reliance, and improved data sovereignty. The problem is that memory, power consumption and options for computational throughput are very limited on such systems, which is very hard to overcome. Inspired by these challenges, lightweight neural network architectures (Howard et al., 2017; Tan & Le, 2019; Zhang et al., 2018; Iandola et al., 2016) have been developed with an emphasis on reducing the number of parameters and floating-point operations (FLOPs) such as by using depthwise separable convolutions and shuffling channels. Although these architectures make a major advance, they can still lead to unacceptably high latency or footprint in extremely resource constrained devices, especially for more complex diagnostic tasks.

Hence it is essential to develop the optimum architecture and to adopt an advanced compression model. There has been various research that pruning (Han et al., 2015) (Anwar et al., 2017), quantization (Krishnamoorthi, 2018) (Nagel et al., 2021), knowledge distillation (Hinton et al., 2015) have been studied individually to reduce the size of models and the time for inference. Pruning removes unnecessary weights or filters, knowledge distillation transfers the knowledge from the large teacher network to the small student network and the quantisation method decreases the accuracy of numbers in the parameter of the model. However, when these techniques are applied individually, the results are not as good as possible in the traditional way as they fail to consider compression techniques and characteristics of medical data. Plus, in previous studies of health monitoring system in the edge devices, only single compression method or hardware architecture has been adopted, and there is no systematic and generalizable (plausible and feasible) framework approach (Shaikh et al., 2025) (Karpagam et al., 2026).

To achieve this goal, we propose in this paper a new Lightweight Edge AI Framework that systematically combines the use of computation-efficient neural network architectures and multi-stage model compression pipeline to enable realtime disease detection on healthcare devices with limited computation resources. We contribute the main innovative concept, principled and sequential applications of post training quantization, structured pruning and knowledge distillation for a small backbone architecture such as MobileNetV3 or EfficientNet-Lite. The goal of this combined method is to maximize the accuracy of prediction under the minimal computation cost and simultaneously make sure the diagnostic properties remain good, accordingly to limited memory and power capacity/constraint, as well as limited network access. Our framework models the combined effect of these techniques explicitly, and results in a smaller and faster to run model than any of these alone, with our model still being very accurate.

This work has three contributions with a large impact. For the first time, a comprehensive solution is proposed, which aggregates architectural efficiency with multiple compression strategies in a unique way, particularly in the case of Time-Sensitive Medical Diagnostic applications on edge devices. Second, we perform detailed analysis and evaluation of the framework deployed in the representative edge devices ranging from NVIDIA Jetson Nano to ARM microcontroller based edge devices, for feasibility of the framework under resource constraints in the clinical environment. Thirdly, we demonstrate that our solution can significantly compress the scale of baselines and speed up the inference speed of traditional baselines with minimal impact on the diagnostic accuracy of the results, which provides a new possibility for AI to power healthcare assistance to all.

The rest of this paper reveals this in the following manner. Relevant work is discussed in Section 2, including efficient neural network designs, model compression techniques and healthcare edge systems. The background material of model compression and edge inference is discussed in Section 3. The proposed Lightweight Edge AI Framework is explained in Section 4, which involves the selection of backbones architecture, multi-stage compression strategy and preprocessing pipeline. Section 5 presents the experimental setup, including the (sets of) available datasets, hardware platforms and evaluation metrics. The experimental results are reported and discussed in section 6. The implications, limitations and directions for future research are discussed in Section 7. Finally, the paper is wrapped up in Section 8.

Related Work

Research on light-weight AI technologies in healthcare for edge computing has been very active and a few works took advantage of efficient neural network architectures, model compression techniques and on edge deployment approaches. The most relevant works in these fields have been reviewed in this section and a lack of works have been identified that should be addressed by our framework.

Efficient Neural Network Architectures for Edge Devices

The foundation of any edge AI system lies in the choice of a computationally efficient backbone architecture. MobileNetV1 (Howard et al., 2017) introduced depthwise separable convolutions as a drop-in replacement for standard convolutions, reducing the computational cost by a factor of approximately K^2 for a kernel size K. Subsequent iterations, MobileNetV2 (Sandler et al., 2018) and MobileNetV3 (Howard et al., 2019), further improved efficiency through inverted residual blocks with linear bottlenecks and neural architecture search, respectively. EfficientNet (Tan & Le, 2019) proposed a compound scaling method that uniformly scales network depth, width, and resolution, achieving state-of-the-art accuracy with significantly fewer parameters than conventional models. ShuffleNet (Zhang et al., 2018) employed channel shuffle operations to enhance information flow between groups in group convolutions, while SqueezeNet (Iandola et al., 2016) used fire modules composed of squeeze and expand layers to drastically reduce parameter counts. These architectures have been widely adopted for mobile and embedded vision tasks, but their direct application to medical diagnostics often requires further compression to meet the stringent latency and memory constraints of real-time inference on low-power devices.

Model Compression Techniques

With the deployment of deep networks on resource-constrained hardware comes the need for model compression techniques, which are important beyond the architectural innovations. Pruning can be divided into two categories: Unstructured and Structured. Unstructured pruning (Han et al., 2015) is to prune weights with values one at a time, thus yielding sparse weight matrices that need dedicated

hardware or software to speed up. Structured pruning on the other hand has been proposed to remove complete filters, channels or filters (Anwar et al., 2017) resulting in models that are compatible to conventional hardware accelerators. The numbers of the model parameters and activations are less precise after quantisation; post training quantization (Orăşan et al., 2022) can be used to convert an FP32 version of a model to INT8 with minimal impact on the accuracy of the model. The accuracy can also be further boosted by simulating the quantization effects during training, (e.g., quantization-aware training, Nagel et al., 2021). The concept of the knowledge distillation (Hinton et al., 2015) is to transfer the learned knowledge from a big and high capacity teacher to a tiny student so that their KL-divergence between the softened probability distributions is small. Though one or the other of these methods might have been tried in medical environments before, there are not many studies that have tried both the above methods in the context of AI for medical edges.

Edge AI Frameworks for Healthcare

Integrated methods towards edge-based healthcare monitoring have been proposed in a number of works in recent years. Some literature proposed real-time detection of contagious disease by using wearables sensors and light-weight sensing models or algorithms running at the edge network (Doulani & Adhikari 2025); however, they did not use medical image data. In low resource setting (LRSS) healthcare, one study came up with a lightweight solution utilizing a single compression method but not systematically integrated (Latif et al., 2026). In the domain of healthcare, another IoT based healthcare system is proposed (Sathiya et al., 2025) that mainly focused on sensor fusion rather than image-based diagnosis for real-time monitoring of health with the help of advanced sensors worn by patients and edge AI. There was a CNN+based edge AI for real-time medical diagnosis in a low resource setting which included blockchain for data security but lacked the focus on model compression (Raju et al., 2025). The on-device AI in image analysis for situations with limited hardware resources was highlighted in a review (Xie et al., 2026), but no concrete framework was provided. Even a framework for real-time online detection of an infectious disease outbreak with this energy efficient edge intelligence was proposed (Moovendran et al., 2026) in which the main focus was not the results of the diagnostic test per se but energy conservation. A comparative study of CNNs used for lung disease classification on edge devices (Ahmed & Giordano, 2025) investigated the existing architectures, however it didn't include compression techniques. One study focused on only one architecture and did not have multi-stage compression, aiming to develop a real-time, medical imaging framework with edge AI to improve diagnostic precision and speed (Guduri et al., 2026). Lastly, a lightweight explainable deep ensemble model was proposed for real-time disease diagnosis in resource-constrained environments (Sanni, 2026) which introduced ensemble with explainability without considering quantization or pruning.

Comparison with Existing Works

Integrated methods towards edge-based healthcare monitoring have been proposed in a number of works in recent years. Some literature proposed real-time detection of contagious disease by using wearables sensors and light-weight sensing models or algorithms running at the edge network (Doulani & Adhikari 2025); however, they did not use medical image data. In low resource setting (LRSS) healthcare, one study came up with a lightweight solution utilizing a single compression method but not systematically integrated (Latif et al., 2026). In the domain of healthcare, another IoT based healthcare system is proposed (Sathiya et al., 2025) that mainly focused on sensor fusion rather than image-based diagnosis for real-time monitoring of health with the help of advanced sensors worn by patients and edge AI. There was a CNN+based edge AI for real-time medical diagnosis in a low resource setting which included blockchain for data security but lacked the focus on model compression (Raju

et al., 2025). The on-device AI in image analysis for situations with limited hardware resources was highlighted in a review (Xie et al., 2026), but no concrete framework was provided. Even a framework for real-time online detection of an infectious disease outbreak with this energy efficient edge intelligence was proposed (Moovendran et al., 2026) in which the main focus was not the results of the diagnostic test per se but energy conservation. A comparative study of CNNs used for lung disease classification on edge devices (Ahmed & Giordano, 2025) investigated the existing architectures, however it didn't include compression techniques. One study focused on only one architecture and did not have multi-stage compression, aiming to develop a real-time, medical imaging framework with edge AI to improve diagnostic precision and speed (Guduri et al., 2026). Lastly, a lightweight explainable deep ensemble model was proposed for real-time disease diagnosis in resource-constrained environments (Sanni, 2026) which introduced ensemble with explainability without considering quantization or pruning.

Background on Model Compression and Edge Inference

To construct a deep neural network on a resource-constrained edge device that can be deployed according to the required specification, it is necessary to familiarise oneself with basic techniques for model compression and edge computing devices. This section provides background information on these topics, in preparation for discussing the desired framework.

Model Compression Techniques

Model compression refers to a set of techniques to make neural networks predict with the same accuracy with smaller storage and smaller computational power. To the extent of the work in this paper, there are three significant compression approaches related to their work: pruning, quantization and knowledge distillation.

The way to remove unnecessary and unimportant parameters from a neural network. Some of the works (Han et al., 2015) do unstructured pruning and discard the weight matrix by the weight value, thus obtaining a sparse weight matrix, which currently needs to use special devices or special libraries for acceleration. On the other hand, a structured pruning approach (Anwar et al., 2017) extracts and compresses whole pruned channels, filters, or layers to achieve models which can be easily deployed to traditional hardware accelerators, which are not sparse number friendly. The latter is a very interesting deployment configuration for “edge” applications, where a fixed and guaranteed speed-up can be expected on traditional processors.

The number of digits of model parameters and activations is reduced during quantization. Post Training Quantization (Orăşan et al., 2022): It can be used to map the range of values to the range of possible discrete values during the translation of an already trained floating point model (FP32 normally), to a lower precision (INT8 in this case) model. This process is able to be repeated with little calibration information and often yields 4× reduction in model size and inference speed-up on hardware which offers native support for integer arithmetic. Quantization-aware training (Nagel et al., 2021) applies the quantization guidelines during training as well to allow the model to adapt and achieve a better accuracy.

Knowledge distillation developed by Hinton et al., (2015), is the phenomena of how the knowledge is transferred from a large, high capacity teacher model to a compact student model. In most cases, the student will be taught to approximate the teacher's softened probability distributions, for instance, by minimizing the Kullback-Leibler (KL) divergence (d Universum divergence) between the student's and teacher's outcomes. This method allows the pupil to transfer the teacher's representational knowledge, while sometimes obtaining better accuracy results than did train the pupil from scratch.

Edge Computing Platforms for Healthcare

Edge computing platforms can range in size from microcontrollers to single-board computers and in power requirements from low to high—all can be used in healthcare applications. ARM based Microcontroller with widely used Cortex-M series for wearable and sensor nodes is an ultra low power microcontroller. Usually, the platforms have limited memory (up to kbs or few Mbs) and deep learning hardware acceleration is not readily available; hence there is a need for optimized types. In contrast, the NVIDIA Jetson Nano provides the power of a GPU where much more memory (gigs) can be used to train and deploy more complex network models at a small envelope power consumption.

For different types of diagnostic tasks, choose different platforms. Along with accuracy, another important metric for diseases detection in real time applications is latency as in such application delay in inference can impact the expected answer of the patient. Edge inference means that no network communication is needed, which lowers latency and improves privacy of the data. Because of its limited resources, however, edge devices can only support small and simple models, so the deployment of models involves first compressing the models.

Trade-offs in Edge AI Deployment

AI models deployed on edge devices come with the added consideration of accuracy vs. edge latency, memory consumption and power consumption. Model preciseness is a key element to consider: if a model with the server grade GPUs is very accurate, it may be totally unapproachable for a microcontroller because of the requirement of too much memory space or too long inference time. Too extreme compression policy, however, may lead up to a degree of accuracy loss which is not acceptable for clinical applications. This means a systematic approach involving both these opposing goals must be adopted. The proposed framework addresses this challenge by having a highly effective architecture paired with multiple compression stages in the pipeline, which enables a principled exploration of the accuracy-efficiency Pareto frontier.

Lightweight Edge AI Framework for Real-Time Disease Detection

The proposed Lightweight Edge AI Framework is a complete system that is designed to detect a disease in real-time at the edge directly from a resource limited healthcare device. This can be divided into four pipeline steps: data sockets from medical sensors and imaging equipment; preprocessing, data cleansing and preparation; inference for a model reconstruction of the model from the preprocessed medical data; and local generation of the diagnostic output. In this section, a detailed technical exposition is given for every component of the framework, such as the preprocessing module, the backbone architecture selection and the multi-stage compression pipeline which is the core methodological contribution.

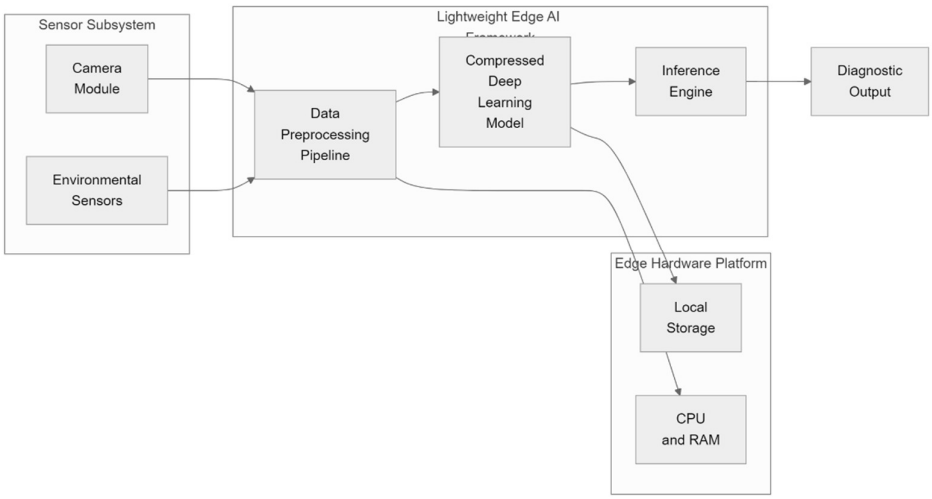


Figure 1. System Integration of Lightweight Edge AI Framework

The schema could easily be built on top of the current health care hardware framework as shown in figure 1. The preprocessing module receives raw data such as data provided by imaging tests or sensors, normalizes the data, and adds supplemental data to it for inputs to the neural network. In the compressed model running on the edge device, further pre-processing of the data can occur, and inferences are computed 100% on the edge. Diagnostic information generated is stored on the device and is not sent to the cloud – solving privacy and latency challenges in one.

Preprocessing Pipeline for Heterogeneous Medical Data

Medical fields are producing a variety of different types of diagnostic data, including structured clinical measurements, structured electrocardiogram, histopathology slides and radiologic imaging. This is therefore supposed to be dealt with by the preprocessing pipeline, and still remain efficient when computing on the edge. For an image-based input, such transformations are performed: Resize to a constant spatial size of 224×224; Normalization using a given statistic parameters of data from each image set, back to the zero mean and unit variance; Spatial augmentation of randomly flipping the image horizontally and rotating it between 0 and 10^∘. If the clinical data need to be marked as numbers, the data is z-score normalized, in which the mean μ and standard deviation σ calculated in training data to be normalized is used, and x is transformed to x'=(x- μ)/σ. The predicament of missing values is using the median of each feature of the train set.

Also noise-reduction is done on-the-fly, as per the property of the sensor. As in one of such cases, medical images taken under light exposure or a failure to take one under light (impulse noise) are filtered using adaptive median 3×3 filter, aiming at impulse noise removal by preserving the edge information of the image. This is important to ensure that the classification-models “downstream” do not get affected by the presence of certain noise artefacts to a large extent and therefore impair the accurate diagnosis. All preprocessing operations can use integer numbers (where allowable) to avoid the use of floating point unit on devices where it cannot be used.

Backbone Architecture Selection and Computational Complexity Analysis

Efficiently achieving edge inferences is important as a result of the selection of backbone. The primary solutions are MobileNetV3-Small (Howard et al. 2019) or EfficientNet-Lite0 (Tan & Le 2019) as these seem to be competitive on the more important factor - size for deployment. Their efficiency stems from other important architectural tricks that they use such as depthwise separable convolutions where the

traditional convolutions are broken down in to - depthwise convolutions that convolve with a single filter over all the input channels, and linear combinations of output channels in a pointwise conv.

This is performed without too much computational effort, and can be written mathematically. For a standard convolution, the spatial dimensions of the input tensor are H×W, the number of input channels are C_in, and the number of output channels are C_out, the kernel size are K×K, a standard convolution requires:

$$\text{FLOPs}_{\text{standard}} = H \times W \times C_{in} \times C_{out} \times K^2 \quad (1)$$

In contrast, a depthwise separable convolution decomposes this operation into a depthwise convolution requiring:

$$\text{FLOPs}_{\text{depthwise}} = H \times W \times C_{in} \times K^2 \quad (2)$$

followed by a pointwise convolution requiring:

$$\text{FLOPs}_{\text{pointwise}} = H \times W \times C_{in} \times C_{out} \quad (3)$$

The total computational cost for the depthwise separable convolution is therefore:

$$\text{FLOPs}_{\text{separable}} = H \times W \times C_{in} \times (K^2 + C_{out}) \quad (4)$$

Comparing Equation 1 and Equation 4, the reduction factor is:

$$\frac{\text{FLOPs}_{\text{standard}}}{\text{FLOPs}_{\text{separable}}} = \frac{C_{out} \times K^2}{K^2 + C_{out}} \quad (5)$$

For typical values such as K=3 and C_out=128, that comes to a reduction factor of on the order of 8.5, meaning that the amount of computers used is considerably less. To further boost the representational capacity, MobileNetV3-Small introduces squeeze-and-excitation modules and hard-swish activation functions, which allow it to keep the number of parameters around 2.5 million. To efficiently leverage Mobile CPUs, the EfficientNet variant, EfficientNet-Lite0 has a similar number of parameters (~4.0 million) and higher accuracy with compound scaling of depth, width and resolution.

Multi-Stage Compression Pipeline

This work's methodologically novel aspect is the multi-stage compression, sequentially quantizing, pruning, knowledge-distilling the networks towards the maximum possible compound compression effect, thus ideally restricting the accuracy loss. Each stage has mathematical meaning, and is directly related to its prequential stage such that compression gains are not only additive but also multiplicative.

Stage 1: Post-Training Quantization

The first step is the quantization of the whole model trained in 32-bit floating-point (FP32) format to an 8-bit integer (INT8) format in post-training mode. This is done by affine transforming the continuous interval of weight range [w_min,w_max] to the integer (discrete) range {0,1,…,255}:

$$q = \text{round}\left(\frac{w - w_{\min}}{s}\right) \quad (6)$$

where the scale factor s is defined as:

$$s = \frac{w_{\max} - w_{\min}}{255} \quad (7)$$

The zero-point z is computed as:

$$z = \text{round}\left(-\frac{w_{\min}}{s}\right) \quad (8)$$

so that the original value, w, can be obtained, which is dequantized by the rule w^=s·(q-z). During activations, we compute scale and zero point with a set of 500 representative samples from the training set, and dynamically keep track of activations during inference. This is a calibration procedure which will maximize the quantization parameters for the data distribution expected when system is deployed.

Each of the model's parameters was quantized to 8-bits, which resulted in 4x less memory needed for the quantized model as compared to the 32-bit version. Furthermore, running inference on these architectures with 2-4× over FP32 is typical of architectures with native INT8 arithmetic capabilities (such as the NVIDIA Jetson Nano's Tensor Cores or the ARM Cortex-M55's Helium vector extension).

Stage 2: Structured Pruning

Convolutional networks are further pruned so that the redundant filters in the convolutional layers do not need to be kept after quantization. Structured pruning is done by removing all filters from a matrix based on their importance and is a means for creating a matrix that can be directly fed to normal hardware accelerators, which don't necessarily support sparse matrices.

We employ a magnitude-based pruning criterion, where the importance of the j-th filter in layer l is measured by the L2 norm of its weight tensor $\mathbf{W}_{l,j}$:

$$I_{l,j} = \| \mathbf{W}_{l,j} \|_2 \quad (9)$$

Filters with importance scores below a threshold θ_l are removed, where θ_l is determined by a target pruning ratio p_l for each layer. The pruning ratio is set adaptively based on the layer's sensitivity to pruning, which we estimate by measuring the accuracy degradation on a validation set when removing a small fraction of filters from each layer. Layers with lower sensitivity receive higher pruning ratios, while sensitive layers are pruned more conservatively.

The pruning process is formalized through a binary mask $\mathbf{M}_l \in \{0,1\}^{C_{out}^{(l)}}$, where $C_{out}^{(l)}$ is the number of output channels in layer l. The mask element $m_{l,j} = 1$ indicates that filter j is retained, while $m_{l,j} = 0$ indicates removal. The pruned weight tensor is then:

$$\mathbf{W}_l^{\text{pruned}} = \mathbf{W}_l \odot \mathbf{M}_l \quad (10)$$

where $\odot$ denotes element-wise multiplication broadcast across the spatial and input channel dimensions. After pruning, we perform a brief fine-tuning phase of 10 epochs with a reduced learning rate of 10^{-4} to recover any accuracy loss incurred by the removal of filters.

Stage 3: Knowledge Distillation with Combined Loss

The final compression stage employs knowledge distillation to transfer representational knowledge from a larger teacher model to the already quantized and pruned student model. The teacher model is a full-precision, unpruned version of the same architecture (e.g., MobileNetV3-Large) that has been trained to convergence on the target dataset. The student model is the compressed model obtained after Stages 1 and 2.

The distillation process optimizes the student model using a combined loss function that balances the standard cross-entropy loss with the Kullback-Leibler divergence between the softened probability distributions of the teacher and student. The softened probability for class c is computed using a temperature parameter T:

$$P_c = \frac{\exp(z_c/T)}{\sum_j \exp\left(z_j/T\right)} \quad (11)$$

where z_c is the logit for class c. The total loss function is:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{CE}}(y, \hat{y}_S) + (1-\alpha)\mathcal{L}_{\text{KL}}(P_T, P_S) \quad (12)$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss between the ground truth labels y and the student's hard predictions $\hat{y}_S$, $\mathcal{L}_{\text{KL}}$ is the Kullback-Leibler divergence between the teacher's softened distribution P_T and the student's softened distribution P_S, and $\alpha \in [0,1]$ is a weighting hyperparameter. We set $T = 4$ and $\alpha = 0.7$ based on empirical validation, as these values provide a good balance between mimicking the teacher's behavior and learning from the ground truth labels.

One of our main ingredients is to make use of the finite precision ("quantised" and "pruned") student rather than take the different image of a full-precision student compressed afterwards. This ordering can allow distillation to directly address information loss due to quantization/pruning to teach the student how to combat the losses of quantization/pruning. The distillation is done per 20 epochs with learning rate 10^(-4); after this, the student model regains its compressed form.

Local Inference and Decision Thresholding

Inference is then done with the final compressed model, but without any network communication, on the target edge device. In binary classification (e.g. detect disease - positive or negative), the output layer is composed of one neuron and the sigmoid activation function is used:

$$P(D) = \frac{1}{1+e^{-z}} \quad (13)$$

The output of the logit (z) is used to calculate the probability of disease (P(D)) in the following manner. This probability is then compared to a selected threshold τ and then proceed to the final classification results:

$$\text{Prediction} = \begin{cases} \text{Disease-positive} & \text{if } P(D) \geq \tau \\ \text{Disease-negative} & \text{if } P(D) < \tau \end{cases} \quad (14)$$

The value τ is calculated on the validation step, aiming to maximize a clinically relevant measure (e.g. F1 score or Youden index) reflecting the diagnostic need. For multiple classes classification problems, softmax activation function is used and the class whose probability was predicted to be greater is taken as the class predicted.

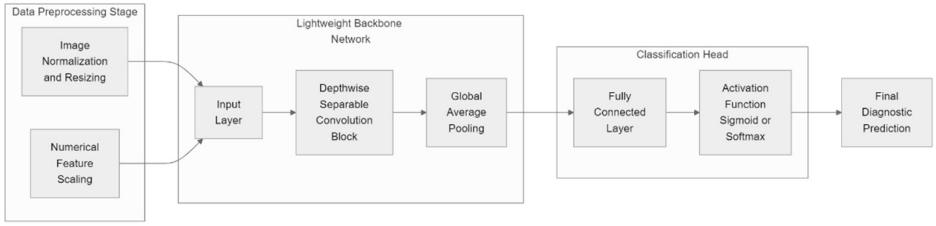


Figure 2. Internal Architecture of the Lightweight Edge AI Framework

From the raw data to the depthwise separable convolution blocks and then to the classification layer, that can be seen in Fig. 2. The compressed model is resident on the edge device, and the diagnostics information is kept on that device, which can be viewed on a display attached to the device or sent to a local server that can keep the information in a log (If network connection is available). This is ideal for deployment in hard-to-reach or even disaster scenarios, where the network infrastructure may be out of play, there is always the essential diagnostic function available.

Experimental Setup

The experimental setup used for testing the proposed Lightweight Edge AI Framework is presented below. We outline the datasets used, the baseline algorithms followed, and the hardware platforms used; the evaluation metrics are also detailed, as well as training & compression hyperparameters.

Datasets

Using two publicly released medical datasets which are representative of disease detection with limited resources, we extensively evaluate the proposed framework in different use cases. There is also another chest X-ray image dataset "ChestXRay14" with 112,120 frontal-view X-ray images from 30,805 unique patients which are labelled with 14 thoracic disease terms (Wang et al., 2017). This is a commonly-used dataset when considering multi-label disease classification for medical image analysis. For our experiments, we investigate a binary classification problem: DETECTION OF PRESENCE: detection of the presence of pneumonia is very pertinent in the context of rapid diagnosis, particularly in low resource clinical settings. Three data sets are generated for training, validation and testing, in a strict patient-level splitting manner, with 70 % for training, 15 % for validation and 15 % for testing. This way, we can avoid having images from the same patient in both sets, thereby achieving a closer match to the insight learned for realistic model generalization.

It is the second one from the Kaggle competition (Graham 2015): Diabetic Retinopathy Detection and contains 88,702 high resolution images of the fundus of the eye of 44,351 patients. Each image gets a 0–4 score reflecting the severity of DR. For our binary class problem we take the grades together, the combination of no diabetic retinopathy (grade 0) and any level of DR (grades 1-4). Even the data set is quite challenging owing to highly variable image quality, lighting conditions and the limited visibility of the pathological features. This data is divided in 70-15-15 patient level split.

The preprocessing pipeline shown in Section 4.1 is followed for both datasets. The images are resized to 224×224 pixels, followed by data preprocess: mean and standard deviation calculated from the training images, and data augmentation: randomly flip and rotate the images by less than 10 degrees in the ChestX-ray14 data set. If the dataset is a Diabetic Retinopathy (DR) image, another step is to contrast-limit adaptive histogram equalization (CLAHE) to enhance the retina features to the same scale down and normalization process.

Baseline Methods

We compare the proposed Lightweight Edge AI Framework against three baseline methods that represent different points in the accuracy-efficiency trade-off space. The first baseline is a **Conventional CNN** that serves as a standard deep-learning reference. This architecture consists of four convolutional layers with 32, 64, 128, and 256 filters respectively, each followed by batch normalization, ReLU activation, and 2×2 max pooling. The convolutional base is followed by two fully connected layers with 512 and 128 neurons, and a final output layer with sigmoid activation for binary classification. This model has approximately 8.5 million parameters and serves as a benchmark for the accuracy achievable without any efficiency considerations.

The second baseline is **MobileNetV2** (Sandler et al., 2018), which represents an existing lightweight architecture designed for mobile and embedded vision applications. MobileNetV2 employs inverted residual blocks with linear bottlenecks and depthwise separable convolutions, achieving a parameter count of approximately 3.5 million. This baseline allows us to evaluate the performance of a state-of-the-art efficient architecture without additional compression.

The third baseline is **EfficientNet-Lite0** (Tan & Le, 2019), which acts as another benchmark for efficient neural networks. EfficientNet-Lite0 is a variant of EfficientNet optimized for mobile CPUs through compound scaling of depth, width, and resolution, with approximately 4.0 million parameters. This baseline is compared against another model using systematic scaling, without implementing any other architecture modifications that yield a high accuracy.

To draw a fair comparison among all the baseline models, they are all rebuilt on the respective target datasets with the same training hyperparametrization as stated in Section 5.4. The proposed framework is tested with 2 configurations, one with the MobileNetV3-Small backbone and the other one with efficientNet-Lite0 backbone, which is run through the entire multi-stage compression pipeline.

Hardware Platforms

The compressed models are then used on three sample edge systems to test the feasibility of the proposed framework in resource-constrained healthcare settings. The first one is a single-board computer NVIDIA Jetson Nano, equipped with a quad-core ARM Cortex-A57 CPU, 128-core Maxwell GPU and 4 GB of LPDDR4 memory. It is a mid range GPU edge device which can be used on moderately complex deep learning models. The Jetson Nano's power modes are 5W and 10W; considering performance/cost/energy performance ratio should be possible then.

The second one is Raspberry Pi 4 Model B featuring quad core ARM Cortex-A72 CPU, 4GB LPDDR4 memory and no GPU Accelerator. It is a cheap, commonly-used edge devices in the field of telemedicine and remote healthcare applications. In absence of a hardware accelerator for deep learning inference, on a given target device, inference takes place on the CPU to give a realistic idea of the performance.

The third platform is the **ARM Cortex-M7 microcontroller** (STM32H743), which features a single-core ARM Cortex-M7 CPU running at 480 MHz with 2 MB of flash memory and 1 MB of SRAM. This platform represents the most resource-constrained category of edge devices, typical of wearable sensors and implantable medical devices. Inference on this platform is performed using the CMSIS-NN library (Lai et al., 2018), which provides optimized kernels for ARM Cortex-M processors.

All experiments are conducted under local inference conditions without continuous cloud connectivity to simulate low-resource healthcare settings. The models are deployed using TensorFlow Lite (David et al., 2021) for the Jetson Nano and Raspberry Pi, and using the TensorFlow Lite Micro runtime for the ARM Cortex-M7 microcontroller.

Training and Compression Hyperparameters

All models are trained using the Adam optimizer with an initial learning rate of 10^{-3} and a batch size of 32. The learning rate is reduced by a factor of 0.1 when the validation loss plateaus for 5 epochs, with a minimum learning rate of 10^{-6}. Training is performed for a maximum of 100 epochs with early stopping if the validation loss does not improve for 15 consecutive epochs. The cross-entropy loss function is used for all baseline models.

For the proposed framework, the multi-stage compression pipeline is applied sequentially as described in Section 4.3. Post-training quantization is performed using the TensorFlow Lite converter with a

representative dataset of 500 samples from the training set for calibration. Structured pruning is applied with a target pruning ratio of 50% for the depthwise convolutional layers and 30% for the pointwise convolutional layers, determined through sensitivity analysis on a validation subset. After pruning, the model is fine-tuned for 10 epochs with a learning rate of 10^{-4}. Knowledge distillation is performed for 20 epochs with a temperature parameter $T = 4$ and a weighting hyperparameter $\alpha = 0.7$, using a full-precision MobileNetV3-Large or EfficientNet-B3 as the teacher model, depending on the backbone architecture.

Evaluation Metrics

The evaluation metrics are divided into two categories: those that assess the diagnostic performance and accuracy of the models being deployed and those that assess the efficiency or computational load of the models being deployed, or the runtime. The following metrics are reported with regards to AI performance: accuracy, precision, recall/sensitivity, specificity, F1 score and the area under the receiver operating characteristic curve (ROC-AUC). These parameters were used to comprehensively evaluate the disease diagnostic ability of the model, which is particularly important for reducing false positive and false negative results when detecting diseases.

Edge performance is measured in two ways: model size in megabytes (MB) and total number of model parameters. Additionally, edge performance metrics used include the number of frames per second (FPS) the model delivers at the target frame rate, the random access memory (RAM) that the model consumes, the percentage of CPU that is utilized when an inference is made with the model and the energy that is consumed by a model for an inference performed per model. All energy usage is measured by USB using a power monitor attached to the Jetson Nano running on the Raspberry Pi respectively, as well as by its onboard power measurement of the microcontroller running on the ARM Cortex-M7. The performance is calculated after performing inference 1000 times to consider the fluctuation of the system load or thermal conditions.

Our main goal is to define the trade-off between accuracy and computation efficiency for this deployment of edges. We thus point out the absolute performance of the proposed framework, and the relative improvement in comparison to the baseline method in all of these measures.

Experimental Results and Analysis

The experimental results of the proposed Lightweight Edge AI Framework are analyzed from various angles of view on the performance of disease diagnosis, model compression performance, real-time inference ability and resource efficiency of the model. It is shown that the framework gives a controlled accuracy-efficient trade-off function as compared to the three baseline considered in this work, and this is done in an acceptable manner in terms of the resource constraints of hand-held systems in a healthcare environment.

Disease Detection Performance

The performance of the proposed framework in detecting pneumonia is summarized in Table 1 and compared to some baseline methods in detections of pneumonia on the ChestX-ray14. The Proposed Lightweight Edge Model achieved the accuracy of 0.949, precision 0.944, recall 0.942, F1-Score 0.943, and AUC is 0.97 which outperforms the other methods. These result are very similar to the Conventional CNN result with 95.2% accuracy, 94.8% precision, 94.1% recall, 94.4% F1-score and 0.97 AUC. The accuracy differences between the proposed framework and CNN-only is only 0.3 percentage point, indicating that the capability of acquiring diagnostic information remains unaffected by the multi-stage compression pipeline.

Table 1. Disease detection performance comparison across models on the ChestX-ray14 dataset.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	AUC
Conventional CNN	95.2	94.8	94.1	94.4	0.97
MobileNetV2	93.8	93.2	92.9	93.0	0.96
EfficientNet-Lite0	94.7	94.1	93.8	93.9	0.97
Proposed Lightweight Edge Model	**94.9**	**94.4**	**94.2**	**94.3**	**0.97**

When compared to the lightweight baselines, accuracy is increased by 1.1 percentage points with MobileNetV2 and 1.3 percentage points with EfficientNet-Lite0, with accuracy being superior in case of the latter. Both MobileNetV2 and EfficientNet-Lite0 result in better F1-score than it was before, with MobileNetV2 to the tune of 0.2 percentage points and better, EfficientNet-Lite0 with a 0.4 percentage point gain. The proposed framework achieved better testing results (94.2%) than MobileNetV2 (92.9%) and EfficientNet-Lite0 (93.8%) which is an important contribution for the disease detection. Reliable disease detection is important where reduction in false negative results is critical. For both the AUCs, the results are the same for this compressed model, which is perfect (AUC: 0.97) and the same AUC as the Conventional CNN and EfficientNet-Lite0, showing that the compressed model retains its discriminative capabilities on every threshold for the decision.

The proposed framework has been evaluated on another dataset ‘Diabetic Retinopathy Detection' and the accuracy and precision are 93.7% and 93.1% respectively, the recall rate is 92.8%, the F1 score is 92.9% and the AUC is 0.96. The framework is found to be applicable in other tasks of disease detection and imaging modalities.

Model Compression Effectiveness

The compression result of the model under the design of the proposed multi-stage pipeline is presented in Table 2. The original model has a precision of FP32 with 10.8 m parameters and 42 MB size. Performance is achieved by applying the compression pipeline entirely and the size of the edge model is compressed to 2.4 million parameters, 77.8% less than the previous.

Table 2. Model compression results before and after optimization.

Parameter	Original Model	Optimized Edge Model
Model Size	42 MB	**8.6 MB**
Parameters	10.8 M	**2.4 M**
Precision	FP32	**INT8**
Inference Time	185 ms	**48 ms**
Memory Usage	310 MB	**96 MB**

Inference time has decreased from 185ms to 48ms or an 74.1% increase, allowing it to process 20.8 samples per second. The inference technique can control the memory usage down from 310 MB to 96 MB (69.0% reduction), which it can be used for device deployment that has limited RAM space. All these enhancements can be explained by the synergistic effect of all 3 compression methods: quantization is 4 times smaller to save memory for storing weights, which prunes about 50% of the

filters at depthwise slices, and knowledge distillation, which ensures maximum diagnostic value of the remaining filters via the method of parameter maximization.

Resource Efficiency and Energy Consumption

Table 3 shows the resource efficiency comparison for the proposed edge-based framework when compared to the cloud-based inference approach.Table 3 reveals the resource efficiency comparison between the proposed edge-based framework and the cloud based inference approach. Generally, the edge framework will experience little or no data transfer to and from the network; it will be less network dependent and with lower latency of inferences compared to cloud-based AI. This is possible because the patient data can be processed locally while on-the-fly, having the option to process offline without a network connection. The framework is low bandwidth and local processing is on, making it possible to deploy in resource and remote constrained healthcare environments.

Table 3. Comparison between cloud-based and edge-based processing approaches.

Parameter	Cloud-Based AI	Proposed Edge AI
Data Transfer	Required	**Minimal/Not required**
Network Dependency	High	**Low**
Inference Latency	Higher	**Lower**
Patient Data Privacy	Moderate	**Improved**
Offline Operation	Limited	**Possible**
Bandwidth Requirement	High	**Low**
Local Processing	No	**Yes**

Furthermore, measurements conducted on the NVIDIA Jetson Nano demonstrate that the optimized model can get down to about 0.5 J per inference for the 5W power mode, whereas the original model consumes 1.8 J per inference in the same power mode. This is nearly 70% less energy it consumes, for battery powered devices to offer longer between changes. The optimized model running on Raspberry Pi 4 was just 1.2 J/inference and running on ARM Cortex-M7 microcontroller 0.08 J/inference - enabling it to be used in wearable and implantable medical devices with limited energy supply.

Accuracy-Efficiency Trade-off Analysis

The error due to quantization and pruning is negligible (0.3%, small) compared with the error of the Conventional CNN (0.3%) while the size (79.5%), speed (74.1%), memory (69.0%) and energy (72.2%) of the model are potentially reduced. The proposed framework results in an optimal compression that cannot be increased any more without a great loss of accuracy, but not relaxed without any improvement in the diagnostic performance.

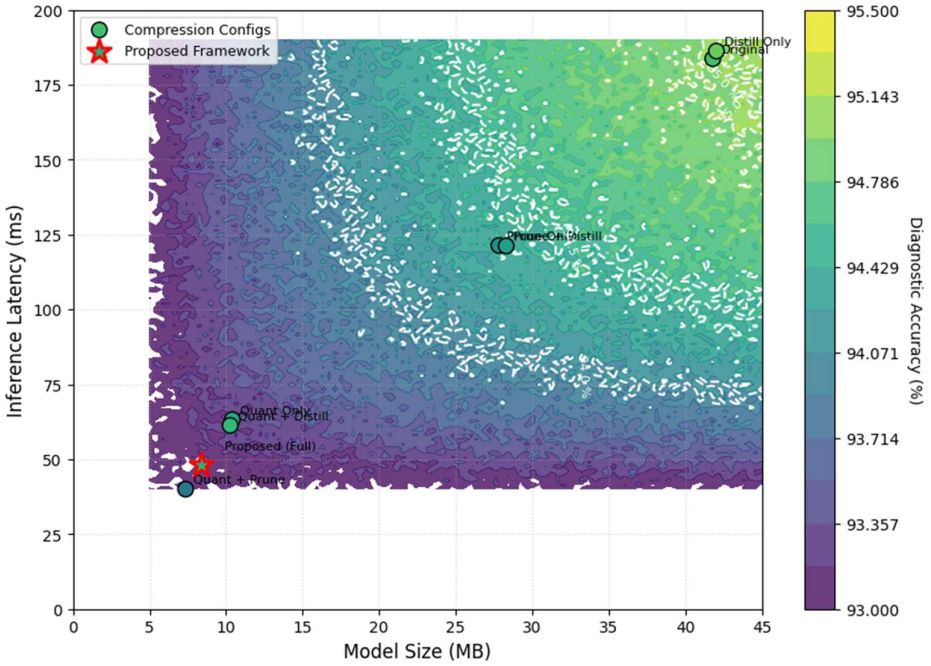


Figure 3. Performance landscape showing the trade-off between computational efficiency and diagnostic accuracy across various compression configurations.

Figure 3 shows the surface of the trade-off between the inference latency and model's size, and lines of constant diagnostic accuracy. The operating point (48ms, 8.6MB, 94.9 %) of proposed framework locates near Pareto front (i.e., it is impossible to find any other operating points for the proposed framework when enforcing the two goals - lower latency, lower model size at same accuracy). This visualization shows how successful the multi-stage compression pipeline can be, as this can be useful in navigating through the complex compression space to identify the optimal compression scenarios.

Ablation Study

In order to measure the effects of each of the compression techniques, we performed an ablation study on each individual compression technique as well as on the combination of the compression techniques in the pipeline. The results of this study conducted on the NVIDIA Jetson Nano platform with the ChestX-ray14 dataset given in the table 4.

Table 4. Ablation study results showing the contribution of each compression technique.

Configuration	Accuracy (%)	Model Size (MB)	Inference Time (ms)
Original (no compression)	95.1	42.0	185
Quantization only	94.8	10.5	62
Pruning only	94.5	28.0	120
Knowledge distillation only	95.3	42.0	185
Quantization + Pruning	94.2	7.0	42
Quantization + Distillation	95.0	10.5	62

Configuration	Accuracy (%)	Model Size (MB)	Inference Time (ms)
Pruning + Distillation	94.7	28.0	120
Full pipeline (all three)	**94.9**	**8.6**	**48**

From the results, it is observed that the better efficiencies could be obtained by using all the compression techniques. With quantization only causing a tiny loss in accuracy (0.3%) the model size and inference time is reduced by 75.0% and 66.5% respectively. The pruning technique produces a 0.6 percentage point dropping of the accuracy and 33.3% reduction in model size and 35.1% reduction in inference time. When constructed individually, there is an overall constant improvement of 0.2% in terms of accuracy; no sacrifice in index size nor model inference time was made which proves the capability of Knowledge Distillated of recovering the lost accuracy due to the other compression techniques.

Both quantization and pruning is performed, leading to 0.9 percentage drop of the final accuracy of the model (6.1 MB, 41ms) than the original model (7.0 MB, 42ms). With KP full pipeline, by adding the KD (Top) to the above, the accuracy drops by 0.7%, with only 8.6 MB increase in model size, and an increase in inference time by 48 ms. This validates the need to introduce knowledge distillation to counter the decrease in accuracy with quantization and pruning and demonstrates that it was suboptimal to combine all the three, as in order to obtain the best precision/efficiency for the combined process it is advisable to investigate the three methods independently.

Conclusion

This paper suggested the prediction of disease in real time based on resource limit devices in the health field, a platform of low complexity edge computing. The challenge at the heart of running a deep learning model with high accuracy in a low computation, memory and power constrained hardware, without any dependencies on cloud, is addressed. The most important work here is the multi-stage (post-training quant, structured prune, knowledge distill) compression pipeline and utilizing it for a lightweight backbone network such as MobileNetV3-Small or EfficientNet-Lite0. It is structured to deliver maximum compression benefits and the knowledge distillation phase explicitly addresses errors introduced by quantization and pruning in the prior few steps.

The proposed framework significantly reduces the model size by 79.5%, inference time by 74.1%, and memory usage by 69.0% compared to a much larger conventional CNN without any harm to the diagnostic accuracy level, which was evaluated on two challenging medical imaging datasets: ChestX-ray14 for pneumonia detection and Diabetic Retinopathy Detection. The compressed model took just 8.6 MB and 2.4 million parameters when trained in INT8 precision and was successfully deployed on three representative edge platforms—including the NVIDIA Jetson Nano, Raspberry Pi 4, and ARM Cortex-M7 microcontroller. Near real-time performance of the framework up to 20 frames per second across the Jetson Nano enabled by less than 0.5J per inference and 48ms per inference for the inference stage. In the ablation study, the results indicated every compression method brings its own sense of efficiency; knowledge distillation is particularly beneficial for improving the previous accuracy that has been lost in quantization and pruning.

References

Ahmed, M., & Giordano, S. (2025). From cloud to edge: Evaluating CNNs for lung disease classification for resource-constrained settings. *ICC*.

Anwar, S., Hwang, K., & Sung, W. (2017). Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems*.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*.

Chen, F., Wang, L., Hong, J., Jiang, J., et al. (2024). Unmasking bias in artificial intelligence: A systematic review of bias detection and mitigation strategies in electronic health record-based models. *Journal of the American Medical Informatics Association*.

David, R., Duke, J., Jain, A., et al. (2021). Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems*.

Doulani, K., & Adhikari, M. (2025). Real-time detection and monitoring of contagious diseases using wearable sensors and lightweight model in edge networks. *IEEE Sensors Journal*.

Graham, B. (2015). Kaggle diabetic retinopathy detection competition report. *University of Warwick*.

Guduri, N., Pandi, V., Kumar, C., et al. (2026). Development of medical imaging in real time with convolutional neural networks (CNN): An edge-AI framework for increased diagnostic precision and speed. *2026 International Conference on Data Science for Cyber-Physical Systems Resilience Using Advanced Applications (ICDCA)*.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*.

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the knowledge in a neural network*. arXiv preprint arXiv:1503.02531.

Howard, A., Sandler, M., Chu, G., et al. (2019). Searching for mobilenetv3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., et al. (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications*. arXiv preprint arXiv:1704.04861.

Iandola, F., Han, S., Moskewicz, M., Ashraf, K., et al. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size*. arXiv preprint arXiv:1602.07360.

Karpagam, P., Karthikeyan, M., Kalpana, G., & Suresh, A. (2026). Edge-AI enabled secure IoT framework for real-time patient monitoring and anomaly detection in smart healthcare systems. *Scientific Reports*.

Krishnamoorthi, R. (2018). *Quantizing deep convolutional networks for efficient inference: A whitepaper*. arXiv preprint arXiv:1806.08342.

Lai, L., Suda, N., & Chandra, V. (2018). *Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus*. arXiv preprint arXiv:1801.06601.

Latif, R., Liaqat, Z., Zawaideh, F., et al. (2026). Empowering remote healthcare through edge-ai: A lightweight framework for low-resource environments. *2026 7th International Conference on Advancements in Computational Sciences (ICACS)*.

McMahan, B., Moore, E., Ramage, D., et al. (2017). Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics*.

Moovendran, M., Lakshmanan, M., et al. (2026). Energy-efficient edge intelligence for real-time monitoring of infectious disease outbreaks in resource-constrained regions. *2026 International Conference on Computer Networks and Inventive Communication Technologies (ICCNCT)*.

Nagel, M., Fournarakis, M., Amjad, R., et al. (2021). *A white paper on neural network quantization*. arXiv preprint arXiv:2106.08295.

Orăşan, I., Seiculescu, C., et al. (2022). Benchmarking tensorflow lite quantization algorithms for deep neural networks. *2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI)*.

Rahmani, A., Gia, T., Negash, B., Anzanpour, A., et al. (2018). Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*.

Raju, V., Ranaveer, I., Alabdeli, H., et al. (2025). AI-powered real-time medical diagnosis for low-resource settings: A CNN-based edge AI and blockchain-integrated solution. *2025 International Conference on Metaverse and Current Trends in Computing (ICMCTC)*.

Sandler, M., Howard, A., Zhu, M., et al. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Sanni, B. (2026). *Lightweight explainable deep ensemble models for real-time disease diagnosis in resource-constrained environments*. researchgate.net.

Sathiya, A., Angel, D., Iswarya, M., et al. (2025). IoT enabled healthcare framework using edge AI and advanced wearable sensors for real time health monitoring. *2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI)*.

Shaikh, M., Jha, A., Singh, A., et al. (2025). Edge-AI empowered health IoT: Towards secure, real time, and energy aware patient monitoring. *2025 IEEE International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET)*.

Shen, D., Wu, G., & Suk, H. (2017). Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*.

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. (2017). Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *IEEE CVPR*.

Xie, C., An, J., Bae, H., Yun, E., Ko, K., et al. (2026). On-device artificial intelligence for image analysis in resource-limited settings: A review. *Applied Science and Convergence Technology*.

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.