Optimization of CNNs via Magnitude-Based Pruning with TFMOT

Anny Faria
National College of Ireland - NCI
MSc. in Data Analytics
Dublin, Ireland
x24124770@student.ncirl.ie

***Abstract*—Skin cancer represents the most type of cancer, impacting millions of individuals around the world each year. Dermatoscopic imaging is a technique to diagnose skin lesions that are not visible. It allows for earlier detection of skin cancer and reduces the need for unnecessary biopsies of benign lesions. For the automated diagnosis of skin lesions, convolutional neural networks (CNNs) have been widely applied because of their ability to extract complex visual features from dermatoscopic images. This project aims to optimize the efficiency of a CNN model by reducing its size and inference cost using pruning techniques, specifically magnitude-based pruning with the TensorFlow Model Optimization Toolkit (TFMOT). To evaluate the models, we use the HAM10000 dataset and compare baseline and pruned CNNs in terms of accuracy, precision, recall, F1-score, and computational cost. The results show that pruning significantly reduces the model size and energy consumption.**

***Index Terms*—Convolutional Neural Network, Pruning, TFMOT, Skin Cancer Lesion**

I. Introduction

Cancer is a disease that affects many people, and the treatment received during this period is extremely painful. Detecting cancer in its early stages can offer a chance of cure. More advanced stages are very difficult.

Artificial neural networks have been applied for dermatoscopic imaging to diagnosis common pigmented skin lesions. In 1994, for the first time, dermatoscopic imaging was used to train an artificial neural network to distinguish melanomas. To train diagnostic algorithms based on neural networks, a large number of images is necessary. However, training accurate diagnostic algorithms using neural networks requires a significant amount of dermatoscopic image databases.

To support this need, for two decades, the Department of Dermatology at the Medical University of Vienna (Austria) and the Cliff Rosendahl skin cancer clinic in Queensland, Australia created the HAM10000 database, which has 10,000 dermatoscopic images [1].

Before the popularization of deep learning, the classification of medical images (including skin, lung, retina, etc.) was done with manual feature extraction (shape, color, texture) and traditional classifiers, such as Decision Tree, KNN, Naive Bayes, SVM, and Adaboost [8].

With the advent of deep learning and recent advancements in image classifier algorithms, Convolutional neural networks (CNNs) have become commonly applied for the purpose of object detection, image segmentation, and semantic segmentation replace manual feature engineering in diagnosis classification. They are designed to learn spatial hierarchies of features by extracting essential features by extracting fundamental characteristics in the initial layers and complex patterns in deeper layers. The pioneer CNN model for images classifications was proposed in 1998 by Lecun et. al [2]. Over time, other Convolutional Neural Networks models came up improving diagnosis classification accuracy and accessibility. However, the size of the CNN model was becoming large demanding significant computational cost for training and inference. This led to the necessity of developing a technique to compress the model by reducing model size and computational cost.

Recently, several method of pruning have emerged and gained attention, due to the ability of reduces significantly model size and computational resources. Pruning is an optimization method designed to eliminate unnecessary connections, parameters, or neurons. Connection pruning involves remove weights that are zero or close to zero to reduce the network memory demands. The pruned model can then undergo retraining and may even attain better accuracy than the original model [3]. Two types of pruning techniques have become common during model training like pruning individual weights and pruning entire neural structures. A universally accepted pruning strategy has not emerge yet, as the effectiveness of various methods tends to change. To chose which method is suitable for the model some questions need to be answered such as which parts of the network should be pruned, how to select the specific items within those parts to remove, the approach to re-training, and whether pruning will be executed layer by layer or across multiple layers simultaneously. The optimal pruning approach is often dependent on the specific network architecture (layers, parameters and others), the volume of training data, and the characteristics of the data itself. Magnitude-based pruning is a common method that reduces model memory usage while maintaining the systems accuracy, making it one of the efficient pruning approaches. This paper studies the application of Magnitude-based pruning to investigate the energy consumption, total time, memory usage and emission of CO_2 in a CNN model based on tree sample: split 10%, split 15% and split 20%.

A. Paper Motivation and Contributions

Many compression and optimization methods for convolutional neural networks, such as pruning and quantization, have been widely explored to enable their execution on edge devices. In this study, we investigate the impact of the Magnitude-based pruning technique on the performance of a CNN model trained for skin lesion classification using the HAM10000 dataset. We apply pruning with TFMOT and quantization via TensorFlow Lite (float16) to evaluate the reduction in model size, inference latency, and energy consumption during training. More specifically, we summarize our contribution in this research as follows:

1) *To implement Magnitude-based pruning with TFMOT improve the accuracy, latency, model size, and energy consumption of convolutional neural networks (CNNs) for skin cancer diagnosis using the HAM10000 dataset.*
2) *To evaluate the environmental impact of training a standard CNN model compare to that of a magnitude-pruned CNN model.*
3) *To improve the representation of minority classes.*

B. Outline

This paper is organized as follows: In Section. II we present an overview of the current knowledge that supports the specific research questions. In Section. III we present the deep learning models and generative AI used to conduct research. Section. IV we present the results and evaluation in this work. Section. V we summarize the main results obtained and future work.

II. Related Work

This section present an overview of the current knowledge that supports the specific research questions.

A. Deep Learning for image classification

Several works have been done in medical image classification, but few studies are available in the field of using compression. Most studies with HAM10000 focus on performance (accuracy, F1, etc.). In a recent study, Chao Chen [5], proposed an overview of CNNs applied to medical image classification, covering 149 studies on Convolutional Neural Networks (CNNs), synthesizing advancements in architectures, preprocessing, and optimization methods from 2019 to 2023. This analysis identifies CNNs as a dominant method due to their ability to automatically extract hierarchical features, reducing manual feature engineering and enhancing classification accuracy. Also, summarizing 38 public datasets across 12 medical conditions (lung disease, brain tumours, and diabetic retinopathy). The positive aspect is this study is a comprehensive coverage good for understanding the models and their limitations unlike other reviews, it uses PRISMA criteria and spans all diseases and body regions, offering a unified reference for CNN in medical imaging. Also, present a systematic classification of CNN methods and highlights effective preprocessing techniques and strategies to improve robustness. The negative aspects is challenges inherent to CNNs in medical imaging, like small and imbalanced datasets, lead to overfitting and biased predictions. Low generalization ability due to image heterogeneity (different scanners, hospitals, and diseases) and noise. Difficulty detecting small lesions. Lack of explainability, which reduces clinical trust. High computational demands, making deployment on mobile or edge devices difficult.

Gaur et al. [6], presented the processing image samples on a pixel-by-pixel basis using deep convolutional neural networks. The results show that the model was very effective in identifying cancer, achieving an accuracy of 76.1%. This study uses data augmentation that helps address the small dataset issue by applying rotation, scaling, and translation. The negative aspect of this research is the small dataset with 3300 images, which is relatively small for CNN training, increasing overfitting risk.

The method know as "data augmentation' is applied to improve the datasets to be trained in machine learning models. The writers Guergueb and Akhloufi The authors Guergueb and Akhloufi [7], used this data argumentatio method in a collection of over 36,000 images from different databases. The investigation aimed to assess deep learning models for skin lesions. The research compared a total of 20 different models. The results indicated that EfficientNetB7 presented better results compared with all other models, achieving an accuracy rate of 99.1%.

These works in aggregate for an understanding and illustrate significantly evolution and improvements in convolutional neural network models in skin lesion image classifiers.

B. Pruning Technique

Pruning techniques has emerged as an optimization method for deep learning models, and have transformed the field of deep neural networks allowing models to become more compact, improving computational resource without sacrificing performance. This review focuses on the most recent and relevant studies that support the methodological decisions adopted in this project. Ferreira et al. [12] investigated the use of magnitude pruning, quantization, and weight pooling with the TensorFlow Model Optimization Toolkit (TFMOT) to optimize a CNN called GBRAS-Net, used in image steganalysis. The authors demonstrated that these techniques can reduce model size by up to 75% with minimal impact on accuracy, and in some cases, with improved performance. Although the application domain is different focused on detecting hidden messages in images rather than medical diagnosis the presented methodology the application of magnitude pruning, is relevant and can be adapted to medical classification scenarios, such as skin cancer diagnosis with CNNs trained on the HAM10000 dataset. However, the study does not evaluate networks common in the medical field, nor does it use clinical images or healthcare specific metrics.

Research made by Cheng et al. [4] present a comprehensive survey of pruning techniques in deep neural networks, organizing them into categories based on when pruning occurs before, during, or after training, the granularity (structured or unstructured pruning), and integration with other techniques

such as quantization and distillation. The authors emphasize that structured pruning, by removing filters or entire channels, is especially useful for reducing latency and enabling execution on specialized hardware, such as mobile devices. This is directly aligned with the present project, which applies structured pruning to SeparableConv2D layers, aiming to maintain compatibility with conversion to TFLite. Furthermore, the work emphasizes the importance of applying fine-tuning after pruning, a practice also incorporated in this project, as a way to recover performance lost during the compression process. Cheng et al. They also discuss deployment scenarios that benefit from reduced model size and lower memory consumption.

Other studies have showcased the combined compression strategies, Khan et al. [13] analyze three compression techniques of pruning, quantization, and distillation. Although they use datasets such as MNIST and CIFAR10, the authors highlight that combining pruning with quantization can result in substantial gains in model size reduction and acceleration, without significant loss of accuracy. This conclusion supports the strategy adopted in this project, in which a CNN model is first compressed through pruning and then converted to TFLite format with 16-bit floating-point quantization (float16). Also raise concerns about the environmental impact of very large networks, noting that training models like GPT3 emits hundreds of tons of CO_2. Although this project does not work with large-scale models, this concern justifies the inclusion of the Emissions Tracker tool as a way to measure and control the energy impact of different model versions, reinforcing the commitment to sustainable practices.

On the other hand, the use of Magnitude Pruning and TensorFlow Model Optimization Toolkit showcased efficiency. Laurent et al. [14] investigate the use of first- and second-order (Taylor) approximations to estimate parameter importance in unstructured pruning techniques. Although the paper focuses on unstructured pruning, the authors demonstrate that simple criteria such as Magnitude Pruning can be as effective as more sophisticated approaches, especially when followed by a fine-tuning process. This conclusion reinforces the choice made in this work to apply structured pruning using the default TensorFlow Model Optimization Toolkit (TFMOT) wrapper, which uses simplified criteria to select the filters to be removed. Furthermore, the authors point out that preserving the loss function during pruning does not guarantee better final performance, which validates the practical strategy of this project of evaluating the final model with real accuracy and F1-score metrics, rather than just loss metrics.

The process of recovering the models predictive performance after removing weights or filters considered redundant is a crucial part of the pruning technique. The studies of Wang et al. [15] proposes a hybrid pruning strategy that integrates unstructured magnitude-based pruning with a statistical screening method based on F-statistics to guide weight selection. The authors simulate sparse training, pruning individual weights based on magnitude. The method also includes post-pruning fine-tuning across multiple epochs to restore accuracy. The results show that combining magnitude pruning with F-statistics leads to higher pruning rates with minimal accuracy degradation, validating the value of data-driven pruning. Despite the entire methodology be different for this present research, the application of fine-tuning post training led to good results. For that reason this work is directly relevant to the presented methodology, which applies magnitude-based pruning followed by fine-tuning to optimize CNNs for classification.

Recent studies have increasingly emphasized the environmental impact associated with the training and compression of machine learning models. Due to the growth of these tools in various fields of knowledge, some researchers are concerned about their impact on the environment. Concerned about the sustainable use of artificial intelligence. For example, Paula et al. [16] used the CodeCarbon tool to measure CO_2 emissions in models compressed via pruning and quantization, observing 20-32% reductions in energy consumption, with performance maintained between 95-99% accuracy. The eco2AI tool, presented by Budennyy et al. [17], provides an integrated framework for energy and emissions tracking in deep learning experiments, supporting sustainability assessments directly within the experimental cycle.

These aggregated works show the importance of pruning to reducing model size, making CNNs more feasible for image classification diagnosis. Also, these works support the methodological choice including traditional metrics such as latency and model size with sustainability considerations.

III. Deep learning and generative AI Methodology

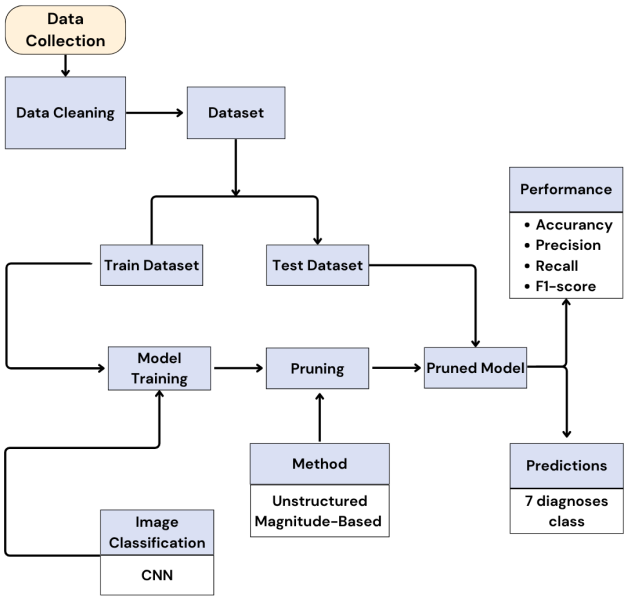


Fig. 1: Project Methodology

This paper uses a Knowledge Discovery-based framework (KDD) method. The main goal of this study is pruning convolutional neural network CNN for skin cancer diagnosis classification using the public HAM10000 dermatoscopic

image dataset which has been referenced in the data collection section. To addresses our research question we prepared a diagram as illustrated in Figure 1. It includes data collecting, exploratory analysis, data preprocessing, model implementation, models training, model pruning and evaluations.

A. Data Collection

The dataset chosen for this research consists of publicly available images of skin lesions dataset HAM10000 from the Harvard Dataverse [1], which contains 10015 data of skin lesions split in two parts folders (HAM10000_images_part_1 and HAM10000_images_part_2) in .jpeg format and fall into 7 classes of skin lesions. Also the dataset has a metadata in .csv format that has all the details of individual images.

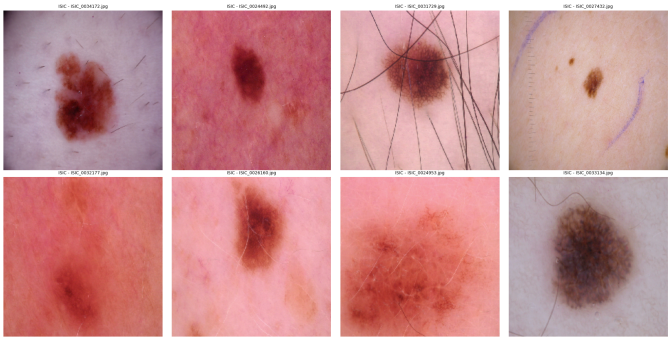


Fig. 2: Sample images of HAM10000 dataset.

B. Exploratory Data Analysis

Exploratory data analysis was done to gain a better understanding of the dataset and metadata. Several findings uncovered during the analysis were essential for processing the images.

For the initial exploratory analysis, we first investigated the dataset that has all the details of individual images, such as age, sex, a unique identifier for a specific type of lesion, a unique identification number for an image, the diagnostic type for technical validation, the geographical location of the skin lesion, and a classification of skin lesions that can aid in diagnosing a condition. The problem of this dataset is it presents 57 missing values in age, 57 "unknown" values in sex and some of classes (dx) are very unbalanced.

Table I shows all the variables present in metadata.

TABLE I: Data Types of the Dataset Columns

Column Name	Data Type
lesion_id	object
image_id	object
dx	object
dx_type	object
age	float64
sex	object
localization	object

As a result of the initial exploratory data analysis, the findings are summarized and illustrated in Figure 3(a)-(f): (a) show the distribution diagnosis of the seven skin lesion categories. The following findings were classified: 6705 melanocytic nevi (nv), 1113 melanomas (mel), 1099 benign keratoses (bkl), 514 basal cell carcinomas (bcc), 327 actinic keratoses (ak), 142 vascular lesions (vasc), and 115 dermatofibromas (df). The dominated class is melanocytic nevi (nv). (b) show the distribution of the age of the patients. The patients were mostly between the ages of 40 and 60. (c) show the distribution of the gender 5406 are male and 4552 are female. (d) Show separate body areas like the back, lower extremities, trunk, upper extremities, abdomen, face, chest, foot, and neck. The following findings show that the major geographical localization diagnosis are in the back and lower extremity. (e) show the distribution of diagnosis by gender. As the result we have male as the major gender with diagnosis, and (f) shows the distribution of diagnosis by age.

C. Data Pre-Processing

Data preprocessing is a crucial steps in deep learning, particularly when working with image data for classification tasks. This process makes the dataset standardized, improves the niceness of the statistics, helps the neural network converge faster, improves numerical stability, eliminates noise, and prevents overfitting. It involves techniques like cleaning, image resizing and batching, normalizing the image pixels and, reducing noise.

For our simple CNN model, preprocessing involved the removal of any missing or invalid image paths to ensure that all samples were valid. Normalization was applied by scaling pixel values to a range of 0 to 1, which standardized the dataset and helped model convergence. The images were resized to 128×128 pixels, matching the models input shape and allowing for reducing time training.

Data Splitting: We can not train a model on a single dataset. So, when we decide to split the data, we should know how many splits of data we need. The HAM10000 dataset was split into training to train the model and enable it to recognize the difficult patterns present in the data, validation subsets to validate our model performance during training, and test. This process of splitting was done three times to evaluate the model under different validation proportions. Initially, the dataset was divided into 90% for training and 5% for validation and 5% Test with a split_ration=10% (validation + test). In subsequent experiments, the split ratios were adjusted to 85/7.5/7.5 witn split_ration=15% and 80/10/10 with split_ration=20% to analyze how different validation sizes affect model performance.

TABLE II: HAM10000 Dataset Split with Different *split_ratio* Values

Split Ratio	Train Sample	Validation Sample	Test Sample
10%	9013	501	501
15%	8512	751	751
20%	8012	1001	1001

Data augmentation: Data augmentation methods play a crucial role in deep learning because enable to treat images applying rotations, flips, scaling, and zooming, which helps the train models to generalize data. These transformations

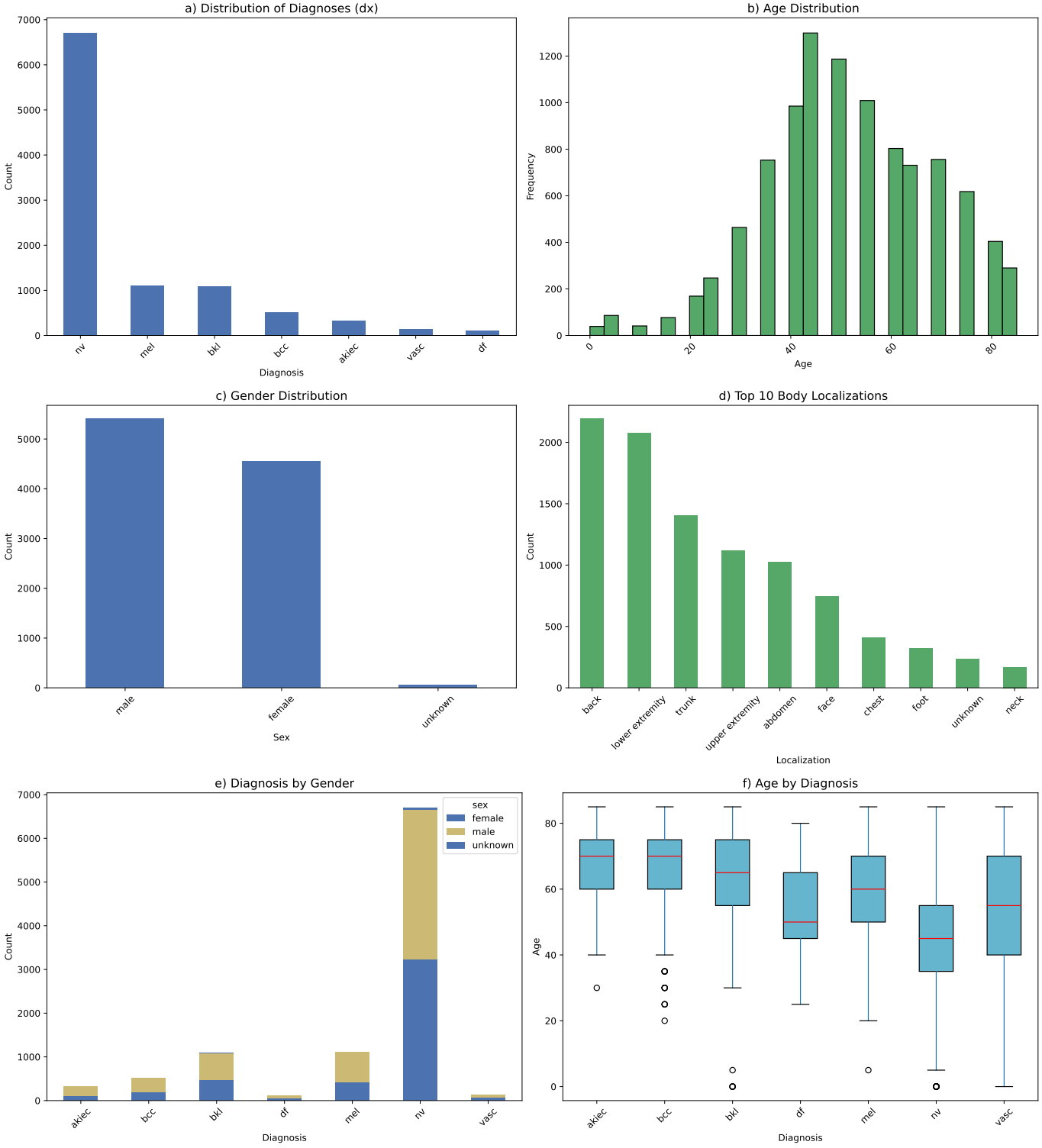


Fig. 3: Exploratory data analyses

were randomly implemented on the training images utilizing a library like TensorFlow/Keras ImageDataGenerator. As shown in figure 4(a)-(d) each image in the HAM10000 dataset was: (a) randomly rotated up to 40 degrees, (b) randomly flipped horizontally, (c) zoomed in or out by up to 20%, and (d) horizontally and vertically shifted by up to 20% of the image dimensions. No synthetic data generation was used. These augmentations were applied during training model to increase data variability and reduce errors.

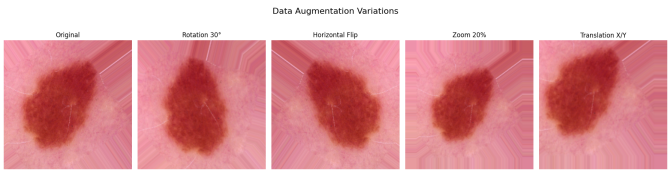


Fig. 4: Data augmentation sample 1

D. Model Implementation

This section presents the implementation of the image classification model for skin lesion analysis using the HAM10000 dataset. The model was developed in Python through Jupyter Notebooks and executed in Google Colaboratory, which offers free GPU resources and simplifies dependency management.

For visualization and metric plotting, the Matplotlib library was employed. The model was implemented using Keras with TensorFlow as the backend. TensorFlow Model Optimization Toolkit (TFMOT) was used to apply pruning for model compression. EmissionsTracker was utilized to monitor energy consumption during training.

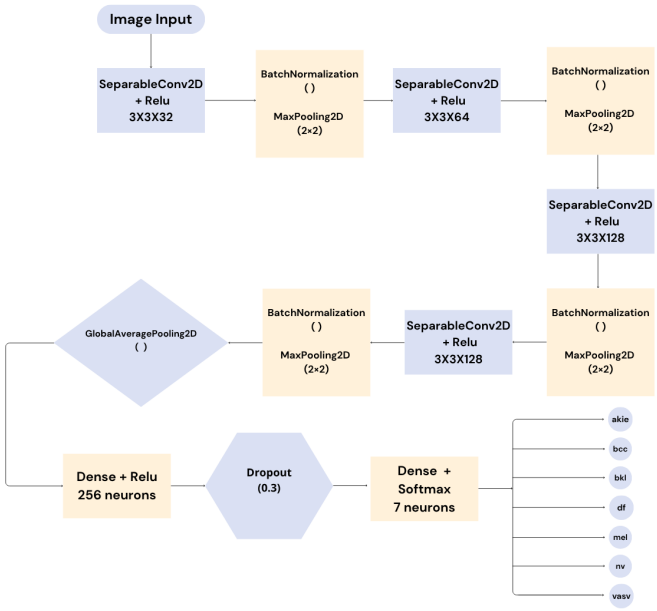


Fig. 5: Architecture of lightweight convolutional neural network.

Figure 5 illustrate the architecture of a *lightweight convolutional neural network* composed of depthwise separable convolutions for efficiency. The model processes dermatoscopic images resized to $128 \times 128 \times 3$, structured as follows:

Convolutional Layers: The network utilizes four convolutional layers, all implemented as depthwise separable convolutions to reduce computational cost while maintaining performance which are mathematically described as:

$$x_d^{(l)}(i,j,c) = \sum_{(u,v)\in\mathcal{K}} W_d^{(l)}(u,v,c) \cdot x^{(l-1)}(i+u, j+v, c) \tag{1}$$

$$x^{(l)}(i,j,k) = \sum_{c=1}^{C_{l-1}} W_p^{(l)}(c,k) \cdot x_d^{(l)}(i,j,c) \tag{2}$$

With $x^{(l-1)}(i+u, j+v, c)$ being the input feature map at position $(i+u, j+v)$ and channel c from the previous layer, $W_d^{(l)}(u,v,c)$ is the depthwise convolution kernel for channel c, $x_d^{(l)}(i,j,c)$ is the output of the depthwise convolution at position (i,j) and channel c, $W_p^{(l)}(c,k)$ is the pointwise (1×1) kernel weight connecting input channel c to output channel k, and $x^{(l)}(i,j,k)$ is the final output of the separable convolution at position (i,j) and output channel k. $\mathcal{K}$ is the spatial domain of the kernel (3×3 window); C_{l-1} is the number of input channels in the previous layer.

Batch Normalization: Batch Normalization is applied after each convolutional layer, to standardize the activations. mathematically it is computed as:

$$\text{BN}(z) = \gamma \cdot \frac{z - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{3}$$

where μ_B and σ_B^2 are the mean and variance over the mini-batch, and γ, β are trainable parameters. This improves convergence during training by stabilizing the distribution of layer inputs and enables increased learning rates.

ReLU: All hidden layers in the CNN are activated using ReLU (Rectified Linear Unit). The ReLU function produces a value of zero for all negative inputs and retains positive values unchanged. This introduces non-linearity into the model and improves learning by allowing only informative signals to propagate while filtering out irrelevant negative activations.

$$\text{ReLU}(z) = \max(0, z) \tag{4}$$

Max Pooling: Each convolutional block is also followed by a Max Pooling operation using a 2×2 filter and a stride of 2.

$$x^{(l+1)}(i,j,k) = \max_{(u,v)\in\mathcal{P}} x^{(l)}(i+u, j+v, k) \tag{5}$$

where $\mathcal{P} = \{0,1\} \times \{0,1\}$. Max pooling reduces the spatial dimensions of feature maps by retaining only the maximum value within each window. This operation helps downsample the features, enhances dominant patterns, and reduces the number of trainable parameters.

Global Average Pooling (GAP):

$$v_k = \frac{1}{H'W'} \sum_{i=1}^{H'} \sum_{j=1}^{W'} x^{(4)}(i,j,k) \tag{6}$$

where H' and W' are the spatial dimensions of the feature map, and $x^{(4)}(i,j,k)$ is the activation from the final convolutional layer.

Dense Layer: After the convolutional and pooling layers, a GlobalAveragePooling2D layer is used instead of flattening. This layer reduces each feature map to a single value by averaging, which helps mitigate overfitting and reduces the model size. The pooled features are then passed to a fully connected Dense layer with 256 neurons and ReLU activation. This layer consolidates the abstracted features for classification.

$$h = \text{ReLU}(\mathbf{W}_1 v + \mathbf{b}_1) \tag{7}$$

Softmax: The final output layer contains 7 neurons, each corresponding to one of the diagnostic categories in the HAM10000 dataset. A Softmax activation function is applied to transform the raw outputs into a normalized probability distribution, allowing for multi-class classification.

$$\hat{y}_i = \frac{\exp((\mathbf{W}_2 h + \mathbf{b}_2)_i)}{\sum_{j=1}^{7} \exp((\mathbf{W}_2 h + \mathbf{b}_2)_j)} \tag{8}$$

These equations represent the complete forward pass of the CNN model for more detail consult [18]–[20].

The complete CNN can be mathematically represented as a composition of operations applied sequentially to the input image $x^{(0)}$:

$$\hat{y}_i = \frac{\exp\left((\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot v + \mathbf{b}_1) + \mathbf{b}_2)_i\right)}{\sum_{j=1}^{7} \exp\left((\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot v + \mathbf{b}_1) + \mathbf{b}_2)_j\right)} \tag{9}$$

where the feature vector v is obtained through four sequential convolutional blocks and a global average pooling operation:

$$v = \text{GAP}\left(f^{(4)}\left(f^{(3)}\left(f^{(2)}\left(f^{(1)}(x^{(0)})\right)\right)\right)\right) \tag{10}$$

Each convolutional block $f^{(l)}$ includes a depthwise separable convolution, followed by batch normalization, ReLU activation, and max pooling:

$$f^{(l)}(x) = \text{MaxPool}\left(\text{ReLU}\left(\text{BN}\left(\text{SepConv}^{(l)}(x)\right)\right)\right) \tag{11}$$

E. Model Training

The model was trained using the Adam optimizer with a learning rate of 5×10^{-4}, chosen for its robustness and adaptive capabilities in handling sparse gradients. Given that the classification task involves seven diagnostic categories, the categorical cross-entropy loss function was used in combination with label smoothing (set to 0.1), which helps to improves generalization. To handle class imbalance in the HAM10000 dataset, class weights were computed based on the frequency of each class and applied during training. This ensures that minority classes are given proportionally higher importance.

Training was conducted for a maximum of 60 epochs. To avoid overfitting and reduce unnecessary computation, two Keras callbacks were employed: EarlyStopping to monitors validation loss and stops training if there is no improvement for 8 successive epochs. The optimal model weights are automatically restored. ReduceLROnPlateau adjusts the learning rate, decreasing it by a factor of 0.5 if the validation loss remains the same for 3 successive epochs, with a minimum limit established at 1×10^{-6}.

F. Model Pruning

To compress the model and reduce inference cost without significantly affecting accuracy, we applied unstructured magnitude-based pruning using the TensorFlow Model Optimization Toolkit (TFMOT). This pruning method zeroes out individual weights with the smallest absolute values, introducing sparsity into the model while preserving its original architecture and layer dimensions. In contrast to structured pruning, which eliminates complete filters or channels, unstructured pruning keeps all neurons and filters intact while deactivates the least important connections between them.

Magnitude-Based Pruning: The model is an unstructured magnitude-based pruning with automatic threshold determination via polynomial decay. The mathematical equation that describe the pruning method are described bellow [1]:

$$s(t) = \begin{cases} 0 & t < 200 \\ 0.5 \cdot \left(\frac{t-200}{800}\right) & 200 \leq t \leq 1000 \\ 0.5 & t > 1000 \end{cases} \tag{12}$$

The polynomial decay schedule was used to gradually increase the sparsity of the model. Specifically, the sparsity was increased from 0% to 50% between steps 200 and 1000. The pruning schedule is defined as: Initial sparsity = 0.0, Final sparsity = 0.5, Begin step = 200, End step = 1000.

Pruning was applied selectively to SeparableConv2D and Dense layers. Other layers, such as pooling and batch normalization, were left untouched. For each weight matrix $\mathbf{W}$ in SeparableConv2D and Dense layers:

$$M_{ij}(t) = \begin{cases} 0 & |W_{ij}| < \tau(t) \\ 1 & |W_{ij}| \geq \tau(t) \end{cases} \tag{13}$$

where W_{ij} = Individual weight at position (i,j) and $\tau(t)$ is the threshold satisfying:

$$\tau(t) = \inf\left\{\tau : \frac{\text{Number of } |W_{ij}| \leq \tau}{\text{Total number of } W_{ij}} \geq s(t)\right\} \tag{14}$$

After pruning, the model was fine-tuned for 20 epochs using a reduced learning rate of 1×10^{-5}, allowing it to adapt to the newly sparse weight structure and recover any worst performance.

Once training was complete, the pruning wrappers were stripped to obtain the final sparse model ready for deployment.

The pruning and fine-tuning pipeline followed these steps:

1) **Model Cloning:** The trained baseline model was cloned, and a pruning wrapper was applied to each trainable SeparableConv2D and Dense layer.

[1]https://stackoverflow.com/questions/60005900/initial-sparsity-parameter-in-sparsity-polynomialdecay-tensorflow-2-0-magnitud

2) **Compilation:** The pruned model was compiled using the same loss function (categorical cross-entropy with label smoothing) and accuracy metric.
3) **Fine-Tuning:** The pruned model was retrained on the training data for 20 epochs using early stopping and learning rate scheduling.
4) **Stripping Pruning Wrappers:** After fine-tuning, the pruning wrappers were removed to obtain the final sparse model.
5) **Evaluation:** The pruned model was evaluated on the test set and compared to the baseline in terms of accuracy, inference latency, model size (in MB), and energy consumption.

IV. Evaluation and Results

To evaluate the performance of the classification models applied in this paper, we used the metrics **Accuracy**, **Precision**, **Recall** and **F1-score**. The formulas used are described below:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{17}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

A. Performance analysis during the training

During the training process, the accuracy metric was recorded at every epoch to assess how effectively the model was learning to classify the input images. At the same time, the categorical cross-entropy loss was computed to quantify how far the predicted probability distributions deviate from the true class labels.

Accurancy: Figure 6 is a comparative plot where displays the behaviour of accuracy as a function of epochs: (a) for the 5% validation dataset and 90% training dataset without pruning. The validation accuracy reaches approximately 70%, higher than the training accuracy around 65%. This suggests that the model generalizes well. (b) for the 5% validation dataset and 90% training dataset with pruning. Accuracy is unstable in the first few epochs and stabilizes around 58–60%, below baseline. There may be underfitting caused by pruning.

Figure 7 the same as showed in figure 6, but the model was training with 7.5% validation and 85% training. (a) validation accuracy reaches approximately 64%, almost equal to training accuracy. This indicates a good fit with a small difference between training and validation. (b) accuracy fluctuates initially but stabilizes around 57%. Also, it is below baseline, suggesting a loss of predictive power with pruning.

Figure 8 the same as showed in figure 6-7, but the model was training with 10% validation and 80% training. (a) good performance and stability. (b) presents instability between epochs 0-10 but then stabilizes.

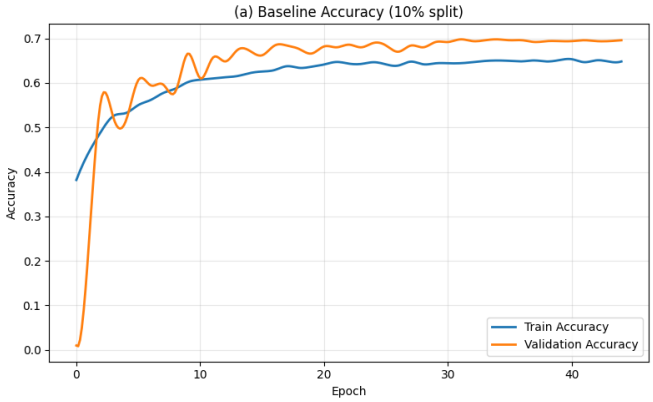
Model Loss: Figure 9 is a comparative plot where displays the behaviour of loss as a function of epochs: (a) for the 5% validation dataset and 90% training dataset, without pruning, the base model attained an loss of about %. Training and validation losses show a gradual and consistent reduction over time. The curves are smooth and well-aligned, indicating stability in the learning process. (b) for the 5% validation dataset and 90% training dataset with pruning. The loss varies at the beginning of training, with an initial peak, but then stabilizes. However, the final loss remains at a higher level, suggesting that the model struggled to learn efficiently.

Figure 10 the same as showed in figure 9, but the model was training with 7.5% validation and 85% training. (a) The both curves decrease continuously, with stable behavior throughout the training. The difference between training and validation is small, demonstrating good generalization capability. (b) The loss curves fluctuate at the beginning of training. After, they stabilize, but the loss remains higher compared to the baseline, suggesting limitations in the pruned models learning.

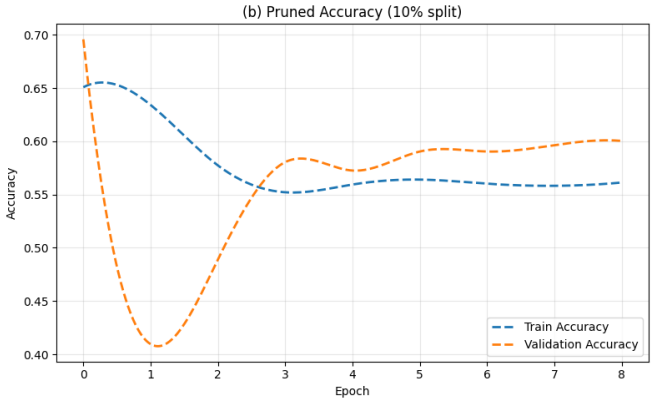
Figure 11 the same as showed in figure 9-10, but the model was training with 10% validation and 80% training. (a) training and validation losses progressively decrease as training progresses. The curves are well-behaved, indicating a robust model with efficient convergence. (b) The validation loss is highly unstable at first, with a sharp increase, but then normalizes. Even after this recovery, the loss remains higher than the baseline model, revealing a lower quality of fit.

Time(s), Energy consumption (J) and Model size (MB): During the training, the time, energy and model size was measured. Figure 12 (a) shows that the pruned models training time is significantly shorter than the baseline models in all validation splits (10%, 15%, and 20%). While the baseline presents high and relatively constant times, the pruned model maintains a much shorter training time, representing a substantial savings. This highlights the computational efficiency of the pruned version, which requires less time to complete the learning process. (b) suggest that the baseline model consumes much more energy during training, regardless of the data split. The pruned model, on the other hand, presents much lower energy consumption, with little variation between the different splits. This difference directly reflects the reduced model complexity, making the pruned version considerably more energy efficient. (c) highlights the difference in model size. The baseline maintains a larger, constant size, while the pruned model is substantially smaller, occupying less space.

CO_2 *emission*: Also, the CO_2 emission was measured. Figure 13 shows the CO_2 emissions (in kg of CO_2 equivalent) of the Baseline and Pruned models as a function of the validation data split (10%, 15% and 20%), with curves smoothed by quadratic regression. The pruned model is more sustainable, with lower and more consistent emissions, regardless of the proportion of validation data. On the other hand, the baseline model, has a greater environmental impact, especially with intermediate divisions.

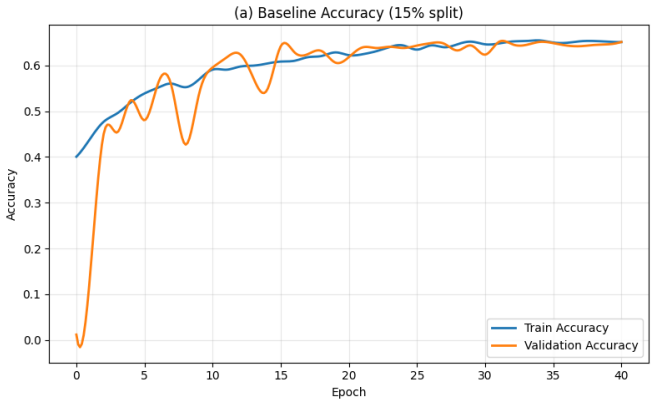


(a) CNN without pruning (split 5%)

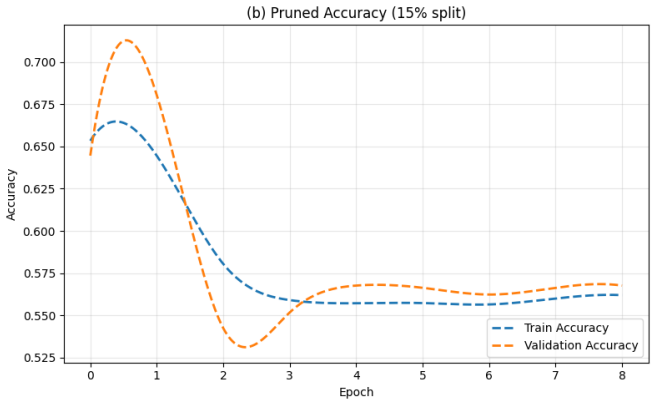


(b) CNN with pruning (split 5%)

Fig. 6: Training and validation performance with and without pruning for the split with validation of 5%.

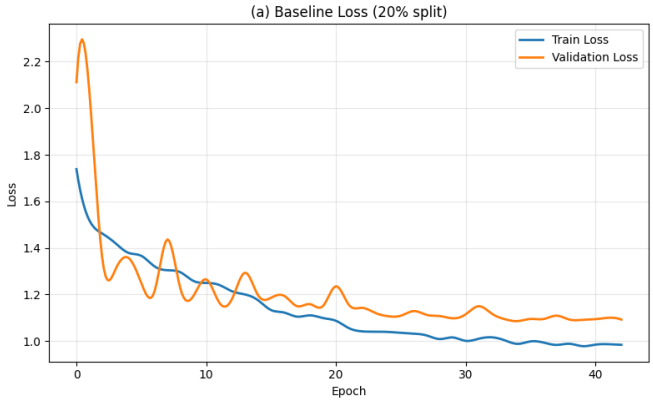


(a) CNN without pruning (split 7.5%)

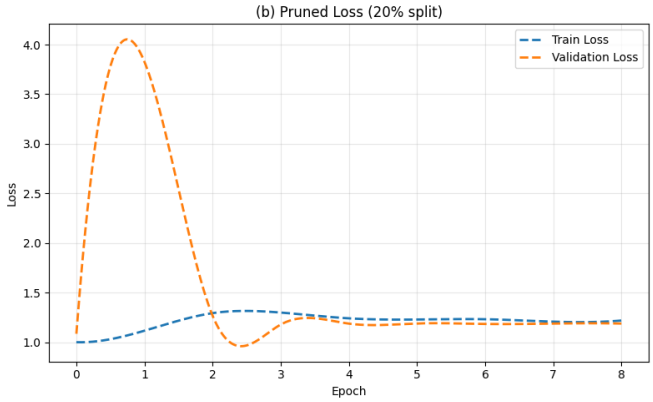


(b) CNN with pruning (split 7.5%)

Fig. 7: Training and validation performance with and without pruning for the split with validation of 7.5%.



(a) CNN without pruning (10% validation split)

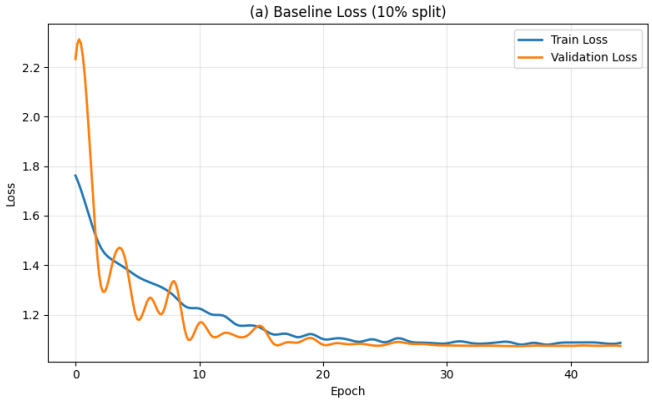


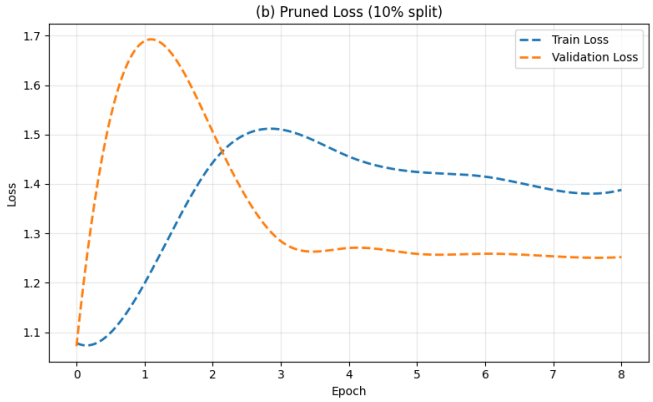
(b) CNN with pruning (10% validation split)

Fig. 8: Training and validation performance of CNN models with and without pruning of 10% validation split.

B. Performance analysis during the testing

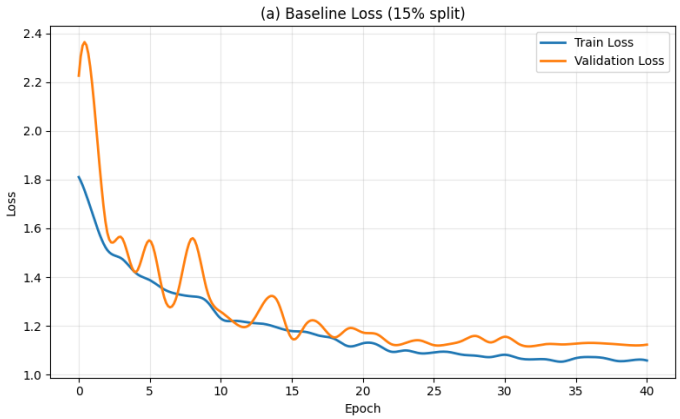
Figure 14 (a) show that the model performed well, with particular emphasis on the “nv”class, which was mostly correct. However, there were significant confusions between similar classes, such as “bkl”, “mel”, and “nv”, indicating difficulty in differentiating similar visual lesions. (b) the behavior is very similar to the baseline. The main difference is the slight improvement in correct predictions for the “mel”class, but

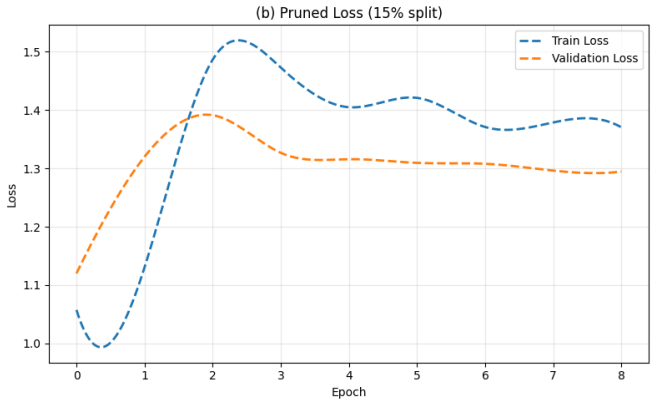




(a) CNN without pruning (5% validation split) (b) CNN with pruning (5% validation split)

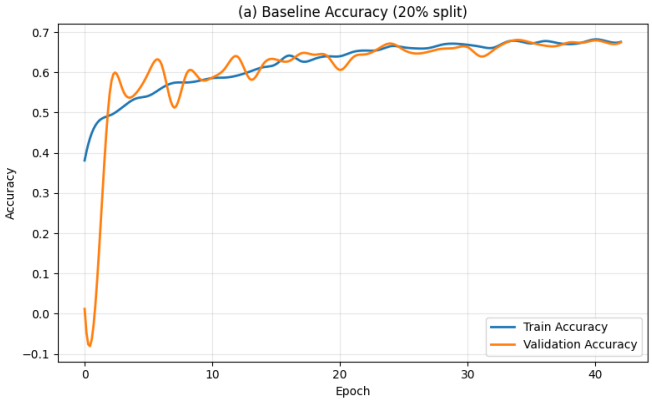
Fig. 9: Training and validation loss performance of CNN models with and without pruning using a 5% validation split.

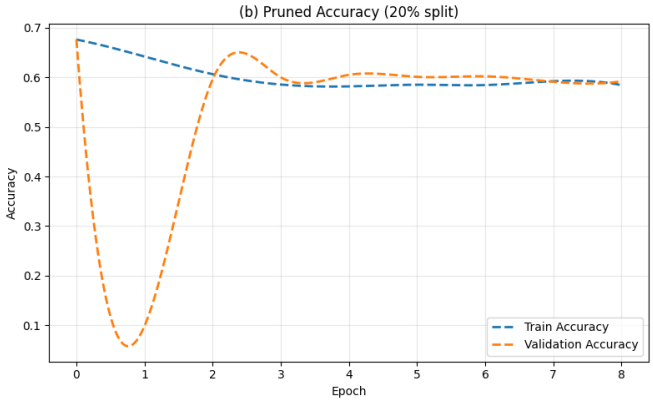




(a) CNN without pruning (7.5% validation split) (b) CNN with pruning (7.5% validation split)

Fig. 10: Training and validation loss performance of CNN models with and without pruning using a 7.5% validation split.





(a) CNN without pruning (10% validation split) (b) CNN with pruning (10% validation split)

Fig. 11: Training and validation loss performance of CNN models with and without pruning using a 10% validation split.

there are still significant confusions in the intermediate classes. There is no significant loss in performance, even with model reduction.

Figure. 15 (a) the model show solid performance, especially for “nv”and “mel”. The “bkl” class shows consistent accuracy, but still shows dispersion to other classes. There is little

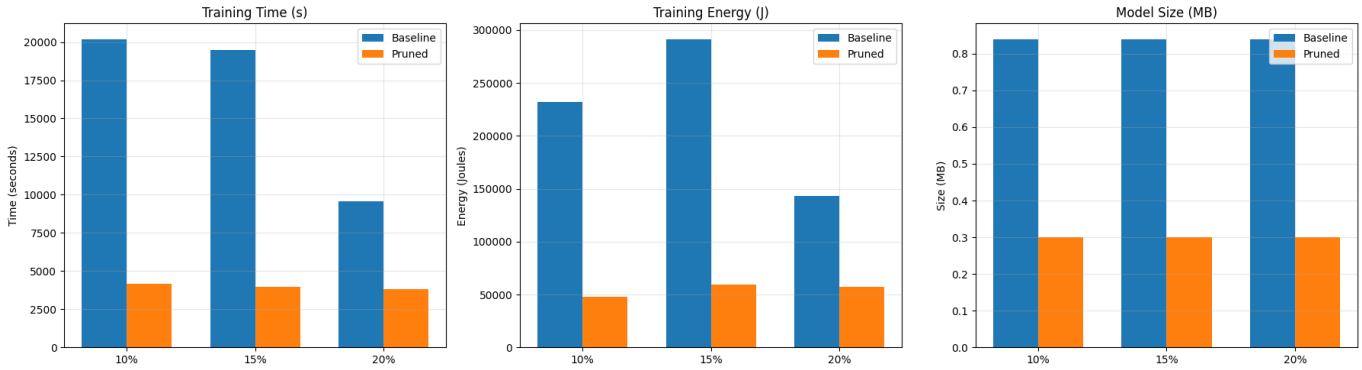


Fig. 12: Comparative analysis of the proposed splits model in relation between (a) training time(s), (b) training energy(J) and, (c) model size(MB).

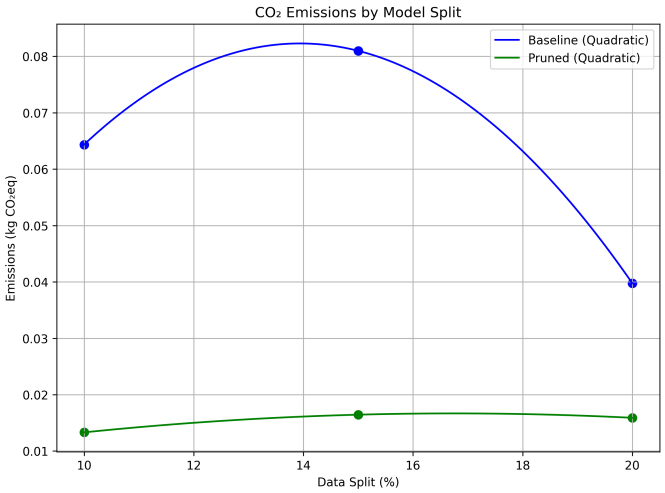


Fig. 13: Comparative analysis of the proposed splits model in relation to CO_2 emission.

confusion in the smaller classes. (b) there was a slight increase in confusion, especially between "bkl", "mel", and "nv", and a small drop in accuracy for the "mel"class. However, overall performance remains acceptable, considering the reduced complexity of the model.

Figure 16 shows that (a) the validation data volume increases, and the model still performs well. The "nv"class continues to dominate in accuracy. However, confusion with "mel"and "bkl" intensifies, revealing a slight impact of the training data reduction. (b) there is greater dispersion in errors, especially in the "nv", "bkl", and "mel"classes.

Some graphs was made to better understand how the model performed.

Figure 17 show that (a) the class "nv" has excellent performance with precision (0.98) and f1-score (0.78), indicating high confidence and consistency. On the other hand, the "df" has high recall (0.67), precision is very low (0.19), resulting in a reduced f1-score (0.30) indicating bad performance.

(b) also the class "nv" maintaining high performance. On the other hand, "df" high recall (0.67) and low precision (0.19), with no improvement from pruning.

Figure 18 show (a) the class "nv" remains a standout, with a high f1-score (0.77) and excellent precision (0.96). On the other hand, "df" class has recall (1.00), but very low precision (0.24), indicating many false positives.

(b) the class "nv" maintains a high f1-score (0.76), solid performance. On the other hand, the class "df" still with very high recall (0.89) and poor precision (0.21), confirming the pattern.

Figure 19 show (a) the class "nv" stands out again, with the highest f1-score (0.82) and good precision (0.91). On the other hand, the class "df" despite high recall (0.64), precision (0.15) is the lowest, resulting in the lowest f1-score (0.23).

(b) the class "nv" remains stable and efficient, with an f1-score (0.77) and excellent precision (0.93). On the other hand, the class "df" recurring behavior with high recall (0.64) and low precision (0.15), making it the most problematic class.

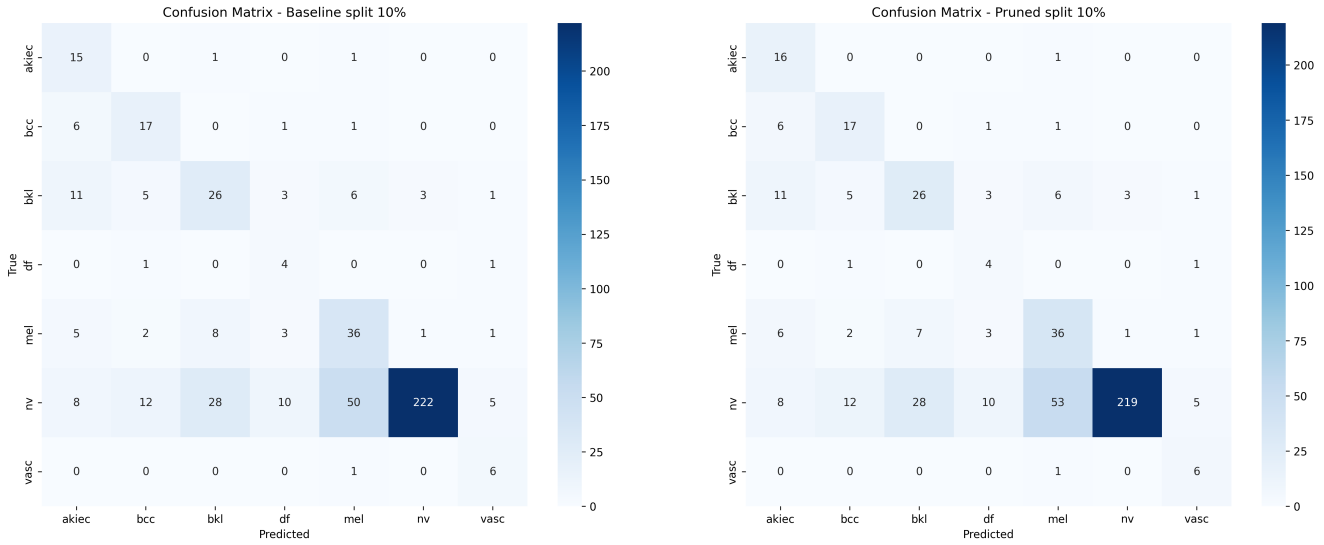
Bellow follow the table representing the classification perform of the model pruned and not pruned by splits.

TABLE III: Classification Report CNN Model Split 10%

Class	Precision	Recall	F1-score	Support
akiec	0.33	0.88	0.48	17
bcc	0.46	0.68	0.55	25
bkl	0.41	0.47	0.44	55
df	0.19	0.67	0.30	6
mel	0.38	0.64	0.48	56
nv	0.98	0.66	0.79	335
vasc	0.43	0.86	0.57	7
Accuracy		0.65		501
Macro Average	0.46	0.69	0.52	
Weighted Average	0.79	0.65	0.69	

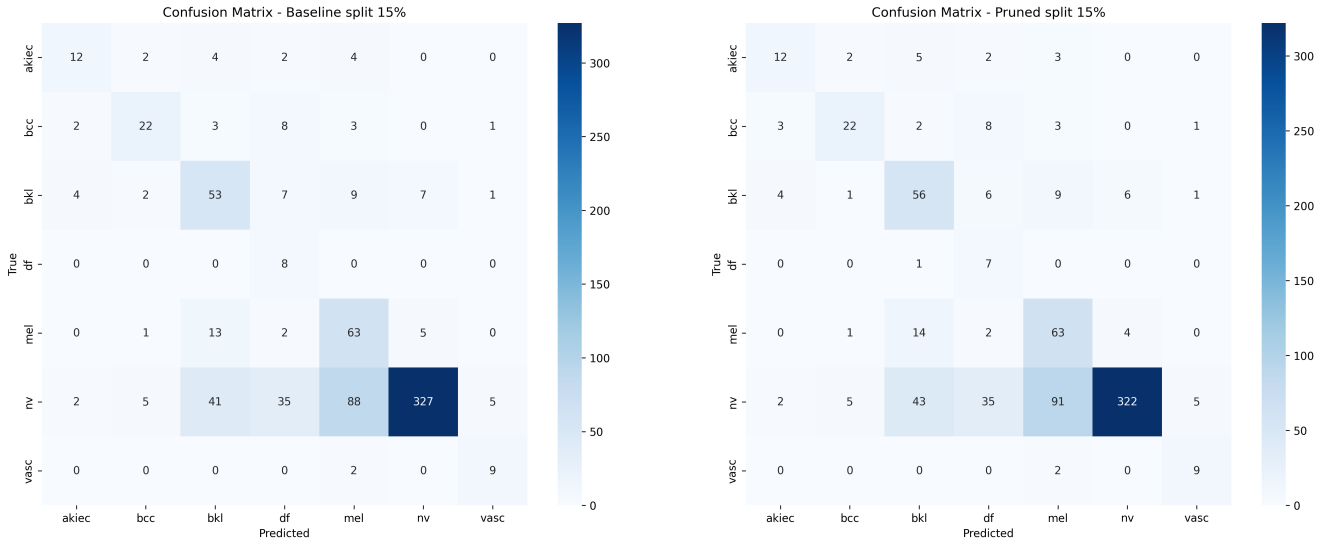
To sum up the performance of the model the models a summary of CNN and Pruned CNN Metrics was provided as shown in table IX bellow.

The table IX shows that the pruned CNN model preserves the predictive performance of the original CNN in all tested scenarios. The 15% split proves to be the most efficient in terms of balancing metrics. Therefore, pruning represents a highly advantageous solution, especially for applications that



(a) CNN (b) CNN Pruned

Fig. 14: Confusion Matrix with split 10%: (a) CNN and (b) CNN Pruned.



(a) CNN (b) CNN Pruned

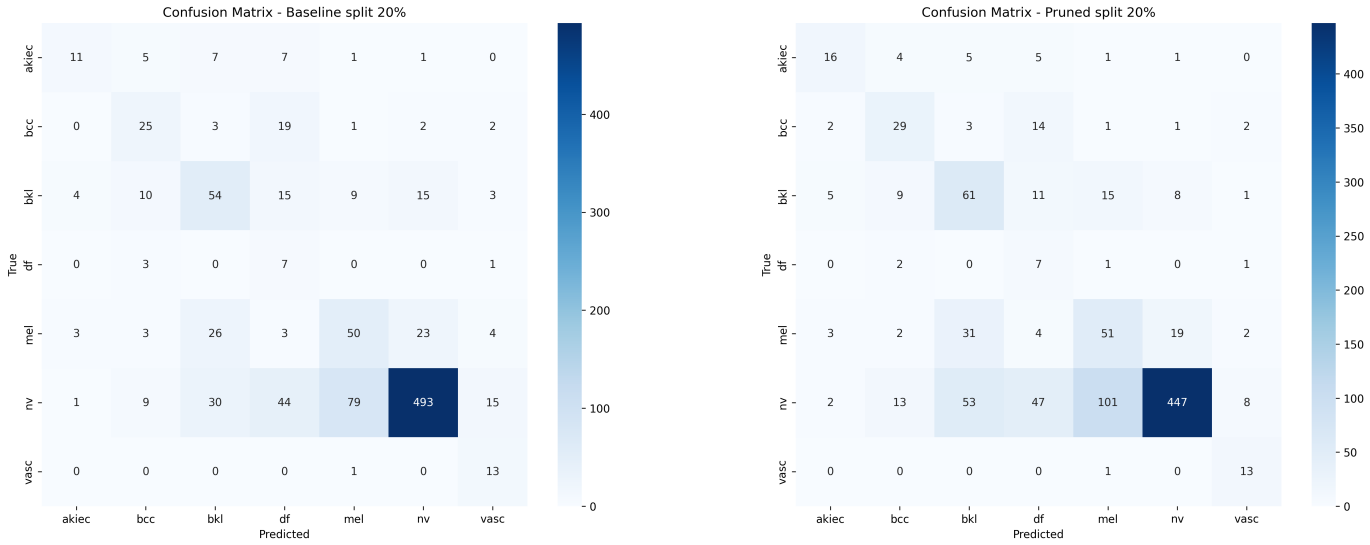
Fig. 15: Same as Fig. 14, with split 15%.

TABLE IV: Classification Report CNN Model Split 15%

Class	Precision	Recall	F1-score	Support
akiec	0.60	0.50	0.55	24
bcc	0.69	0.56	0.62	39
bkl	0.46	0.64	0.54	83
df	0.13	1.00	0.23	8
mel	0.37	0.75	0.50	84
nv	0.96	0.65	0.78	503
vasc	0.56	0.82	0.67	11
Accuracy		0.66		752
Macro Average	0.54	0.70	0.55	
Weighted Average	0.80	0.66	0.70	

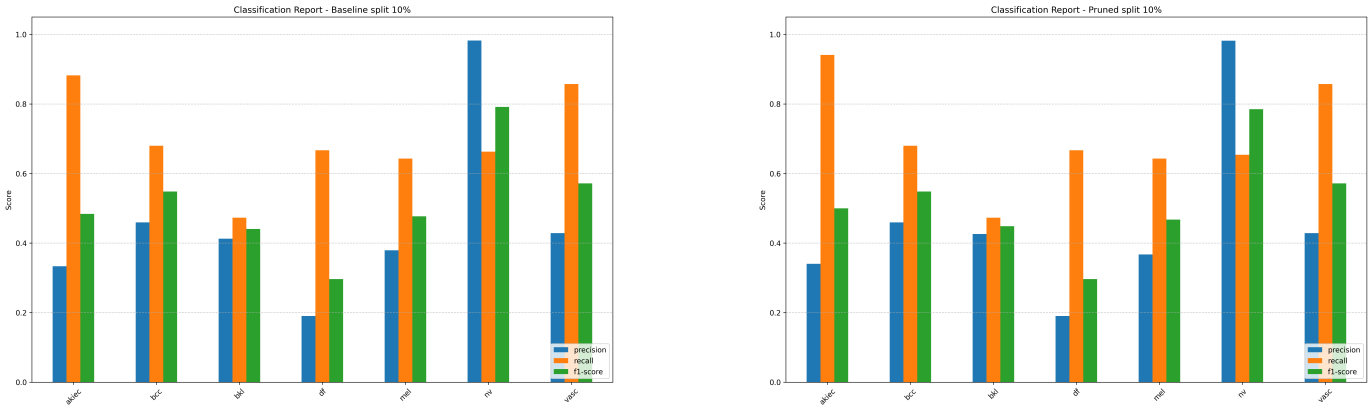
TABLE V: Classification Report CNN Model Split 20%

Class	Precision	Recall	F1-score	Support
akiec	0.58	0.34	0.43	32
bcc	0.45	0.48	0.47	52
bkl	0.45	0.49	0.47	110
df	0.07	0.64	0.13	11
mel	0.35	0.45	0.40	112
nv	0.92	0.73	0.82	671
vasc	0.34	0.93	0.50	14
Accuracy		0.65		1002
Macro Average	0.45	0.58	0.46	
Weighted Average	0.75	0.65	0.69	



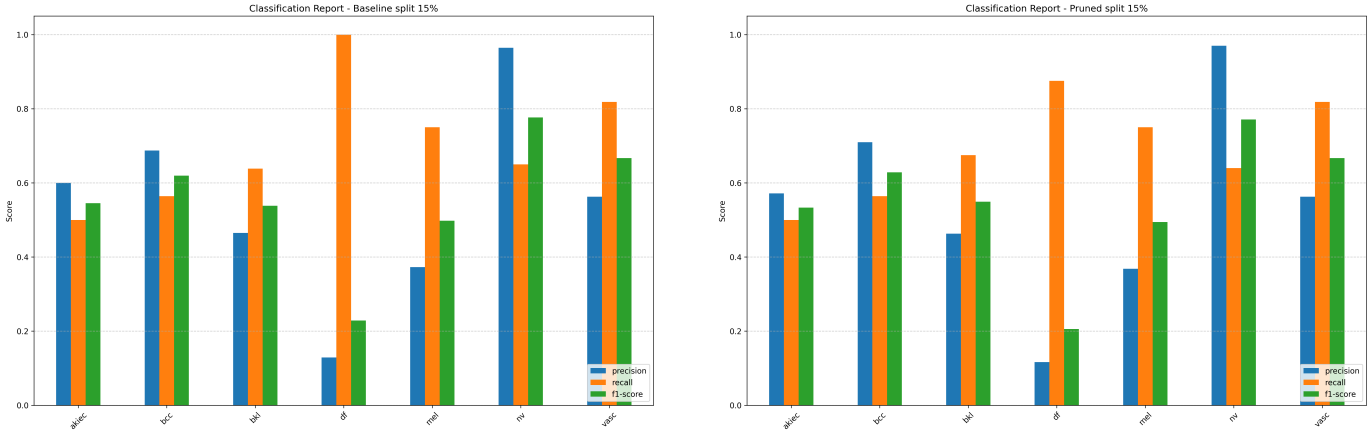
(a) CNN (b) CNN Pruned

Fig. 16: Same as Fig.14 with split 20%.



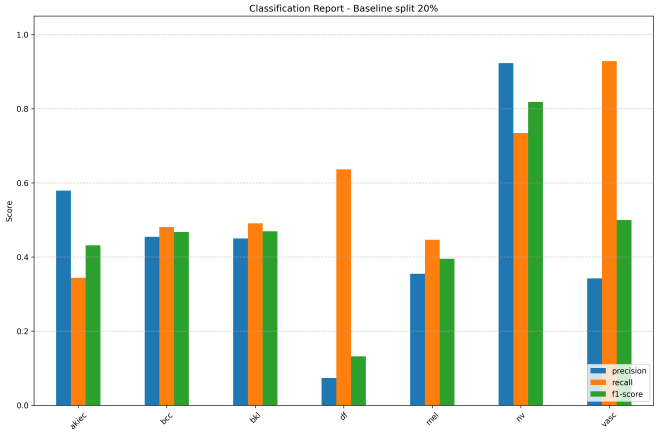
(a) CNN (b) CNN Pruned

Fig. 17: Classification Report Bar Graph of with split 10% (a) CNN and (b) CNN Pruned.

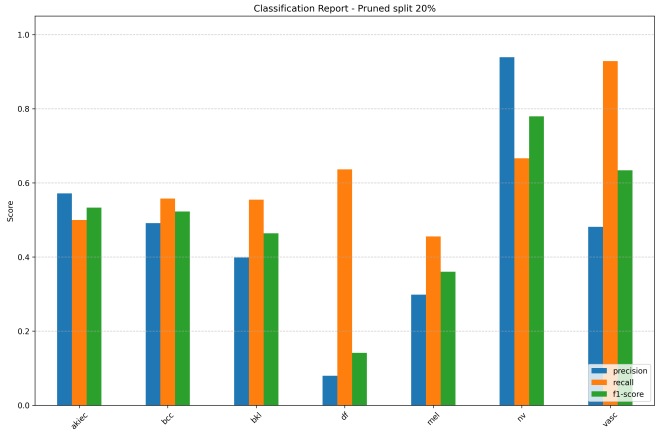


(a) CNN (b) CNN Pruned

Fig. 18: Same as Fig.17 with split 15%.



(a) CNN



(b) CNN Pruned

Fig. 19: Same as Fig.17 with split 20%.

TABLE VI: Classification Report CNN Pruned Split 10%

Class	Precision	Recall	F1-score	Support
akiec	0.34	0.94	0.50	17
bcc	0.46	0.68	0.55	25
bkl	0.43	0.47	0.45	55
df	0.19	0.67	0.30	6
mel	0.37	0.64	0.47	56
nv	0.98	0.65	0.78	335
vasc	0.43	086	0.57	7
Accuracy		0.65		501
Macro avg	0.46	0.70	0.52	
Weighted avg	0.79	0.65	0.68	

TABLE VII: Classification Report CNN Pruned Split 15%

Class	Precision	Recall	F1-score	Support
akiec	0.57	0.50	0.53	24
bcc	0.71	0.56	0.63	39
bkl	0.46	0.67	0.55	83
df	0.12	0.88	0.21	8
mel	0.37	0.75	0.49	84
nv	0.97	0.64	0.77	503
vasc	0.56	0.82	0.67	11
Accuracy		0.65		752
Macro Average	0.54	0.69	0.55	
Weighted Average	0.81	0.65	0.69	

TABLE VIII: Classification Report CNN Pruned Split 20%

Class	Precision	Recall	F1-score	Support
akiec	0.57	0.50	0.53	32
bcc	0.49	0.56	0.52	52
bkl	0.40	0.55	0.46	110
df	0.08	0.64	0.14	11
mel	0.30	0.46	0.36	112
nv	0.94	0.67	0.78	671
vasc	0.48	0.93	0.63	14
Accuracy		0.62		1002
Macro Average	0.47	0.61	0.49	
Weighted Average	0.76	0.62	0.67	

require lightweight and efficient models without significant loss of accuracy.

Figure 20 show the roc curves comparison between CNN and CNN pruned models. As we can see, all models generalise well with similar AUC score.

TABLE IX: Summary of CNN and Pruned CNN Metrics

Model	Accuracy	Precision	Recall	F1-score
CNN (10%)	0.6507	0.7870	0.6507	0.6862
Pruned CNN (10%)	0.6467	0.7873	0.6467	0.6822
CNN (15%)	0.6569	0.8026	0.6569	0.6963
Pruned CNN (15%)	0.6529	0.8055	0.6529	0.6932
CNN (20%)	0.6517	0.7550	0.6517	0.6901
Pruned CNN (20%)	0.6228	0.7573	0.6228	0.6677

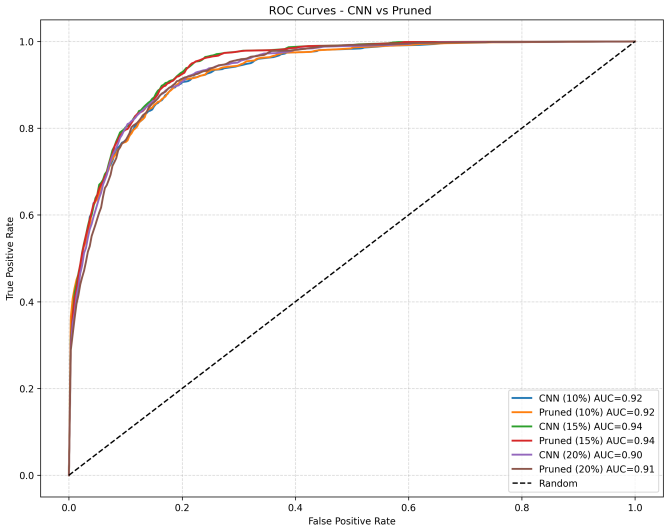


Fig. 20: Roc Curves Comparison Between CNN and CNN Pruned Models.

V. Conclusions and Future Work

This paper has investigated the application of magnitude-based pruning techniques, using the TensorFlow Model Optimization Toolkit (TFMOT) demonstrating that the methodology is effective in reducing the size and computational cost of convolutional neural networks (CNNs) for classifying skin

lesions without significantly compromising accuracy. The results indicate that the pruned models performed comparable to the baseline model in metrics such as accuracy, precision, and F1-score, while also being more efficient in terms of training time, energy consumption, and CO_2 emissions. The 15% data split for validation proved to be the most balanced, balancing predictive performance and computational efficiency.

Finally, this study reinforces the feasibility of adopting compression and optimization strategies in neural networks applied to healthcare, making them more sustainable and accessible, especially on devices with limited resources. Future work proposes investigating other methods to address class imbalance and explore more robust pre-trained models, with a focus on improving performance in minority classes.

REFERENCES

[1] P. Tschandl, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Harvard Dataverse*, vol. 4, 2018. [Online]. Available: https://doi.org/10.7910/DVN/DBW86T

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.

[4] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[5] C. Chen, N. A. M. Isa, and X. Liu, "A review of convolutional neural network based methods for medical image classification," *Comput. Biol. Med.*, vol. 185, p. 109507, 2025.

[6] L. Gaur, U. Bhatia, and S. Bakshi, "Cloud driven framework for skin cancer detection using deep CNN," in *Proc. 2nd Int. Conf. Innovative Practices Technol. Manage. (ICIPTM)*, vol. 2, pp. 460–464, Feb. 2022.

[7] T. Guergueb and M. A. Akhloufi, "Melanoma skin cancer detection using recent deep learning models," in *Proc. 43rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, pp. 3074–3077, Nov. 2021.

[8] M. H. Jafari, S. Samavi, N. Karimi, S. M. R. Soroushmehr, K. Ward, and K. Najarian, "Automatic detection of melanoma using broad extraction of features from digital images," in *Proc. IEEE Eng. Med. Biol. Conf. (EMBC)*, pp. 1357–1360, Aug. 2016.

[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," *arXiv preprint arXiv:1608.08710*, 2016. [Online]. Available: https://www.researchgate.net/publication/307536925_Pruning_Filters_for_Efficient_ConvNets

[10] S. Vadera and S. Ameen, "Methods for pruning deep neural networks," *IEEE Access*, vol. 10, pp. 63280–63300, 2022.

[11] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[12] G. Ferreira, M. H. d. N. Marinho, V. Severo, and F. Madeiro, "Optimization strategies applied to deep learning models for image steganalysis: Application of pruning, quantization and weight clustering," *Appl. Sci.*, vol. 15, no. 9, p. 4632, 2025. [Online]. Available: https://doi.org/10.3390/app15094632

[13] I. Khan and M. Mandal, "Analyzing compression techniques for computer vision," *arXiv preprint arXiv:2305.08075*, 2023. [Online]. Available: https://arxiv.org/abs/2305.08075

[14] C. Laurent, C. Ballas, T. George, N. Ballas, and P. Vincent, "Revisiting loss modelling for unstructured pruning," *arXiv preprint arXiv:2006.12279*, 2020. [Online]. Available: https://arxiv.org/abs/2006.12279

[15] M. Wang, Y. Guo, S. Liu, and Y. Xiao, "Exploring neural network pruning with screening methods," *arXiv preprint arXiv:2502.07189*, 2025. [Online]. Available: https://arxiv.org/abs/2502.07189

[16] E. Paula, J. Soni, H. Upadhyay, *et al.*, "Comparative analysis of model compression techniques for achieving carbon efficient AI," *Sci. Rep.*, vol. 15, p. 23461, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-07821-w

[17] S. Budennyy *et al.*, "Eco2AI: Carbon emissions tracking of machine learning models as the first step towards sustainable AI," *arXiv preprint arXiv:2208.00406*, 2022. [Online]. Available: https://arxiv.org/abs/2208.00406

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[19] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *arXiv preprint arXiv:1610.02357*, 2017. [Online]. Available: https://arxiv.org/abs/1610.02357

[20] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," *arXiv preprint arXiv:1706.03059*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03059