ENHANCED MULTIPLE DENSE LAYER EFFICIENTNET

by

Aswathy Mohan

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

Department of Electrical and Computer Engineering

Indianapolis, Indiana

August 2024

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Mohamed El-Sharkawy, Chair

Department of Electrical and Computer Engineering

Dr. Brian King

Department of Electrical and Computer Engineering

Dr. Maher Rizkalla

Department of Electrical and Computer Engineering

Approved by:

Dr. Brian King

Dedicated to My Parents: Govindan Mohan and Latha Mohan

My husband: Krishnakanth Muraleedhara Kaimal

My grandparents, teachers, friends, family, colleagues and all well-wishers.

ACKNOWLEDGMENTS

As I write this report, I feel like I have learned quite a bit about AI and Deep Learning, a field that was quite new to me when I joined graduate school back in 2021. I am sure words will not do justice to those who put their arms around me as I went through this journey. But, I will try.

First and foremost, I would like to express my sincere gratitude to Dr. Mohamed El-Sharkawy, for his continuous support and mentoring. This is my first experience of conducting thesis research at the highest level and without his guidance, I am positive I wouldn't be where I am today. I would also like to thank Dr. Brian King and Dr. Maher Rizkalla for serving on my thesis committee. Likewise, I want to thank Ms. Sherrie Tucker who selflessly helped me throughout my time at graduate school.

Last but not the least, I am forever indebted to my family for being by my side all this while. It's their unwavering support and unconditional love that propelled me to finish my thesis research on time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

δ	Delta for local gradient
γ	Gamma for focal loss/imbalance
Φ	Phi for activation function
σ	Sigmoid for sigmoid cross-entropy loss
$\sum$	Summation for summing junction
θ	Represents the parameters
η	Learning rate

ABBREVIATIONS

AI	Artificial Intelligence
BNf	Batch Normalization
CAN	Controller Area Network
CNN	Convolutional Neural Network
FLOPs	Floating Point Operations
GAP	Global Average Pooling
GPU	Graphics Processing Unit
MBConv	Mobile Inverted Bottleneck Convolution
NAS	Neural Architecture Search
ReLU	Rectified Linear Unit

ABSTRACT

In the dynamic and ever-evolving landscape of Artificial Intelligence (AI), the domain of deep learning has emerged as a pivotal force, propelling advancements across a broad spectrum of applications, notably in the intricate field of image classification. Image classification, a critical task that involves categorizing images into predefined classes, serves as the backbone for numerous cutting-edge technologies, including but not limited to, automated surveillance, facial recognition systems, and advanced diagnostics in healthcare. Despite the significant strides made in the area, the quest for models that not only excel in accuracy but also demonstrate robust generalization across varied datasets, and maintain resilience against the pitfalls of overfitting, remains a formidable challenge.

EfficientNetB0, a model celebrated for its optimized balance between computational efficiency and accuracy, stands at the forefront of solutions addressing these challenges. However, the nuanced complexities of datasets such as CIFAR-10, characterized by its diverse array of images spanning ten distinct categories, call for specialized adaptations to harness the full potential of such sophisticated architectures. In response, this thesis introduces an optimized version of the EffciientNetB0 architecture, meticulously enhanced with strategic architectural modifications, including the incorporation of an additional Dense layer endowed with 512 units and the strategic use of Dropout regularization. These adjustments are designed to amplify the model's capacity for learning and interpreting complex patterns inherent in the data.

Complimenting these architectural refinements, a nuanced two-phase training methodology is also adopted in the proposed model. This approach commences with the initial phase of training where the base model's pre-trained weights are frozen, thus leveraging the power of transfer learning to secure a solid foundational understanding. The subsequent phase of fine-tuning, characterized by the selective unfreezing of layers, meticulously calibrates the model to the intricacies of the CIFAR-10 dataset. This is further bolstered by the implementation of adaptive learning rate adjustments, ensuring the model's training process is both efficient and responsive to the nuances of the learning curve.

Through a comprehensive suite of evaluations, encompassing accuracy assessments, confusion matrices, and detailed classification reports, the proposed model demonstrates notable improvement in performance. The insights gleaned from this research not only shed light on the mechanisms underpinning successful image classification models but also chart a course for future aimed at bridging the gap between theoretical model and their practical applications. This research encapsulates the iterative process of model enhancement, providing a beacon of future endeavors in the quest for optimal image classification solutions.

1. INTRODUCTION

The developments in deep learning (DL) algorithms for computer vision applications opened the door for eliminating the need to extract classification features [1]. Deep learning has become one of the Turing test generations foundational technologies, continually powering leaps in artificial intelligence, including image classification. Image classification is the process of labeling images according to their content. It underpins crucial applications, such as medical diagnosis and autonomous vehicles. Classification algorithms are called upon to not only be highly accurate, but also to generalize to differing datasets. One of the main hurdles to this generalization lies in the tendency of models to overfit to the training set. Unlike traditional convolutional neural networks (CNNs), EfficientNet employs a systematic scaling method that results in improved efficiency and accuracy at the same time. Amongst all the EfficientNet versions tested, EfficientNetB0 strikes the sweetest spot of an optimal scaling and a high and relatively optimal efficiency. However, adapting such models to specific tasks like CIFAR-10 image classification necessitates further refinement to fully exploit their potential. Recognizing this, an enhanced EfficientNetB0 model is proposed, which incorporates an additional Dense layer and Dropout regularization, complemented by a strategic training approach that includes both initial training with frozen pre-trained weights and subsequent fine-tuning. This research presents a comprehensive overview of the enhanced model, detailing the architectural modifications, training strategy, and evaluation results. The fine-tuning method yields noticeable benefits of classification accuracy and model robustness. These results potentially help the AI community develop future deep learning models such that, when applied to demanding classification tasks using complex data sets, they perform well, without having to look deep into the physical reality for correlations.

1.1 MOTIVATION

Real-world image data is inherently complex and diverse. The CIFAR-10 dataset, comprising images across ten different categories, encapsulates this complexity through its wide array of objects, textures, and scenarios. Traditional models often struggle to capture and generalize the nuanced patterns present in such varied datasets. The motivation of this the-

sis is to push the boundaries of what is achievable with EfficientNetB0, tailoring it to thrive amidst the complexity of real-world data. EfficientNetB0 is renowned for its balance between model size, computational efficiency, and accuracy. However, there remains a substantial opportunity to further bridge the gap between efficiency and accuracy, particularly for specific tasks like CIFAR-10 image classification. By fine-tuning and enhancing the model, this work aims to achieve higher accuracy without significantly compromising on computational demands, making advanced deep learning more accessible and practical for a broader range of applications.

A persistent challenge in deep learning models is overfitting, where the machine learning model gives accurate predictions for training data but not for new data. The motivation of this thesis includes developing strategies, such as the introduction of additional Dense layer, Dropout regularization, and strategic layer freezing, to mitigate overfitting, thereby enhancing the models ability to generalize across different datasets effectively. Transfer learning and fine-tuning represent powerful techniques in deep learning, allowing models to leverage knowledge from one task and apply it to another. The motivation here is to explore the extent to which EfficientNetB0, pre-trained on ImageNet, can be adapted through fine-tuning to excel on CIFAR-10. This approach provides a pathway to rapid, efficient learning and offers insights into the transferability of features across different domains.

1.2 OBJECTIVES OF RESEARCH

One of the primary goals of this thesis is to tackle the tricky issue of overfitting, that is when a model learns the details and noise in the training data to the extent that it negatively impacts its performance on new data. This thesis will look for smart ways to adjust a model's design and training process to prevent this from happening. This way, the model won't just memorize the training data but will become a reliable tool for making sense of new, unseen data, making it much more useful in real-world scenarios. Secondly, balancing the scales between getting the model to be super accurate and keeping it efficient to run is another big focus for this work. To ensure the proposed model is not too heavy or slow to use, it is imperative that its architecture is thoroughly examined. After all, what good is

a state-of-the-art model if it needs an unrealistic amount of computing power to function? The intention here is that the proposed model is accessible to a wide variety of people and organizations, not just those with access to powerful computers. Leveraging what's already known is smart, which is why we're big fans of transfer learning and fine-tuning. By carefully tweaking the EfficientNetB0 model, which has already learned a lot from the ImageNet dataset, it becomes even more adept at recognizing images from the CIFAR-10 dataset. The aim here is to fine-tune the model in such a way that it boosts its performance on CIFAR-10 images, all while holding onto the valuable knowledge it has gained from its previous training. It is also important for fellow researchers to understand how the proposed model makes decisions. So, we're incorporating tools and methods that shed light on the model's thought process. The hope is that by making the model's workings clearer, there will be trust and spark more interest in how these technologies work, both among everyday users and fellow researchers. Lastly, this research is done with an intent to share what has been learnt, the tools that were developed, and the successes and difficulties encountered along the way with the broader AI and deep learning community. By documenting this journey, the hope is to enrich the pool of collective knowledge, encourage others to build on this work and explore new horizons in the field.

2. LITERATURE REVIEW

2.1 Computer Vision

Computer Vision is a domain within Artificial Intelligence (AI) that empowers machines to extract useful information from images and videos. It aims to replicate tasks performed by human visual system with a high level of accuracy. These tasks can be Image Detection, Image Classification, Image Segmentation, Image Enhancement etc. Today, Computer Vision is being employed in numerous applications including but not limited to Automotive [1],[2], Manufacturing [3],[4], Healthcare [5],[6], Surveillance and Security [7],[8]. It is a multidisciplinary field concerned with enabling computers to interpret and understand visual information from the real world. It encompasses a wide range of techniques and methodologies aimed at emulating human vision capabilities using digital images or video sequences as input. Computer vision has a rich history spanning several decades, marked by significant advancements and breakthroughs in both theory and practical applications. The early 2000s witnessed a paradigm shift with the adoption of machine learning techniques, particularly support vector machines (SVMs) and ensemble methods, for various computer vision tasks. The breakthrough success of deep learning, particularly convolutional neural networks (CNNs), in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9], revolutionized the field of computer vision. Models like AlexNet, VGG, and ResNet achieved unprecedented accuracy in image classification tasks. Figure 2.1 visualizes the diversity of the ILSVRC2012 object categories [9].

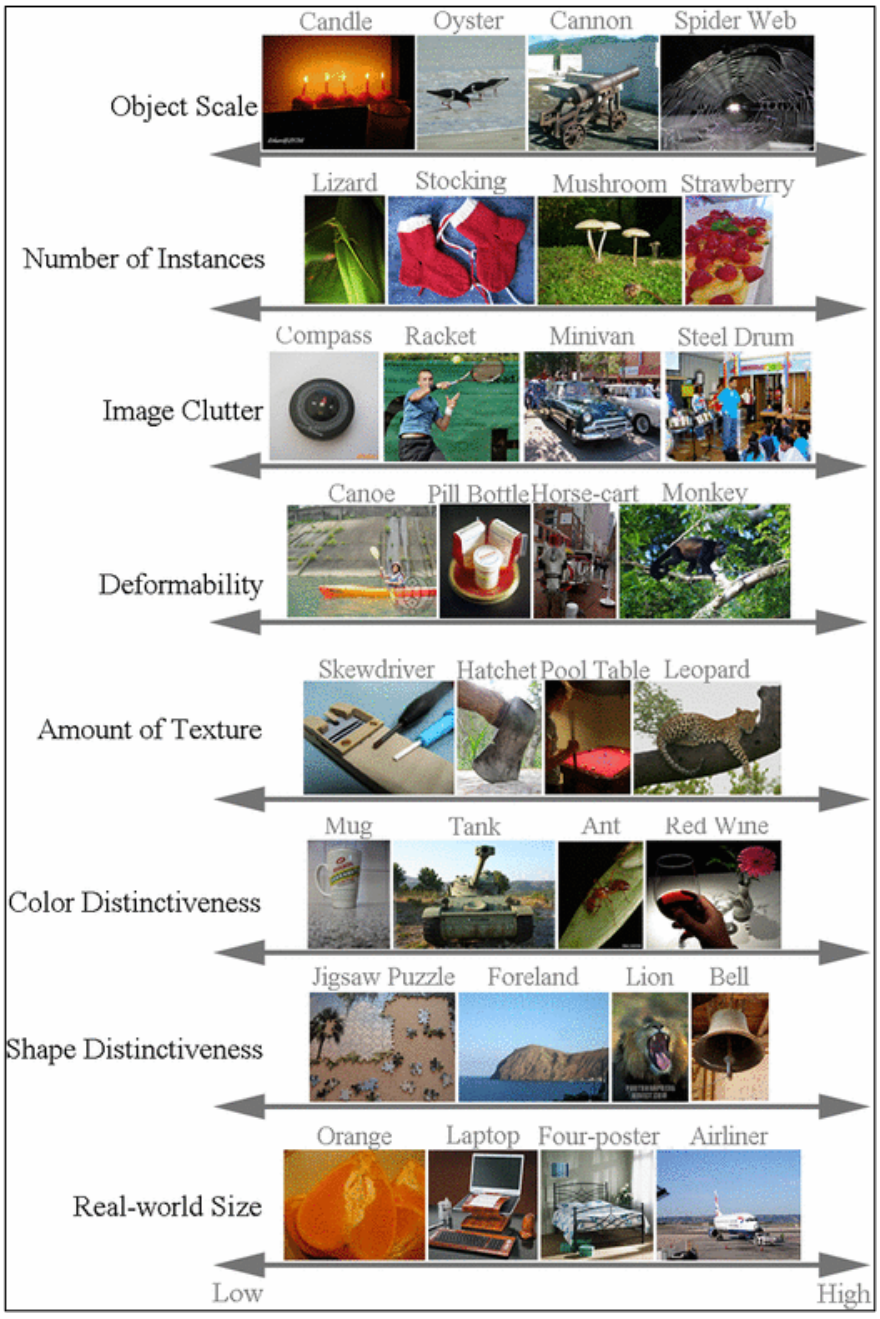


Figure 2.1. Diversity of data in the ILSVRC image classification and single object localization tasks.[9]

2.2 Image Classification

Capturing an image has never been this easy, thanks to widely available technology nowadays. Consequently, there exists a large amount of data in the form of images [10]. Image Classification is an important Computer Vision technique that allows images to be classified according to different pre-defined categories [11]. It plays a vital role in applications such as identifying objects in photographs, autonomous vehicle navigation [12], medical image analysis [13], [14] etc. A sufficient classification system and enough number of training samples are prerequisites for a successful classification [15]. It is a fundamental task in computer vision that involves assigning a label or category to an input image based on its content. This task has been revolutionized by deep learning approaches, particularly convolution neural networks (CNNs), which have demonstrated state of the art performance in various classification benchmarks. The goal of image classification is to develop models that can automatically and accurately assign appropriate labels to unseen images. Traditionally, image classification relied on handcrafted features and shallow machine learning algorithms, such as Support Vector Machines (SVMs) or Random Forests. However, the advent of deep learning, particularly convolutional neural networks (CNNs), has revolutionized image classification, enabling significant advancements in accuracy and performance. Figure 2.2 illustrates different layers of Image classification [16].

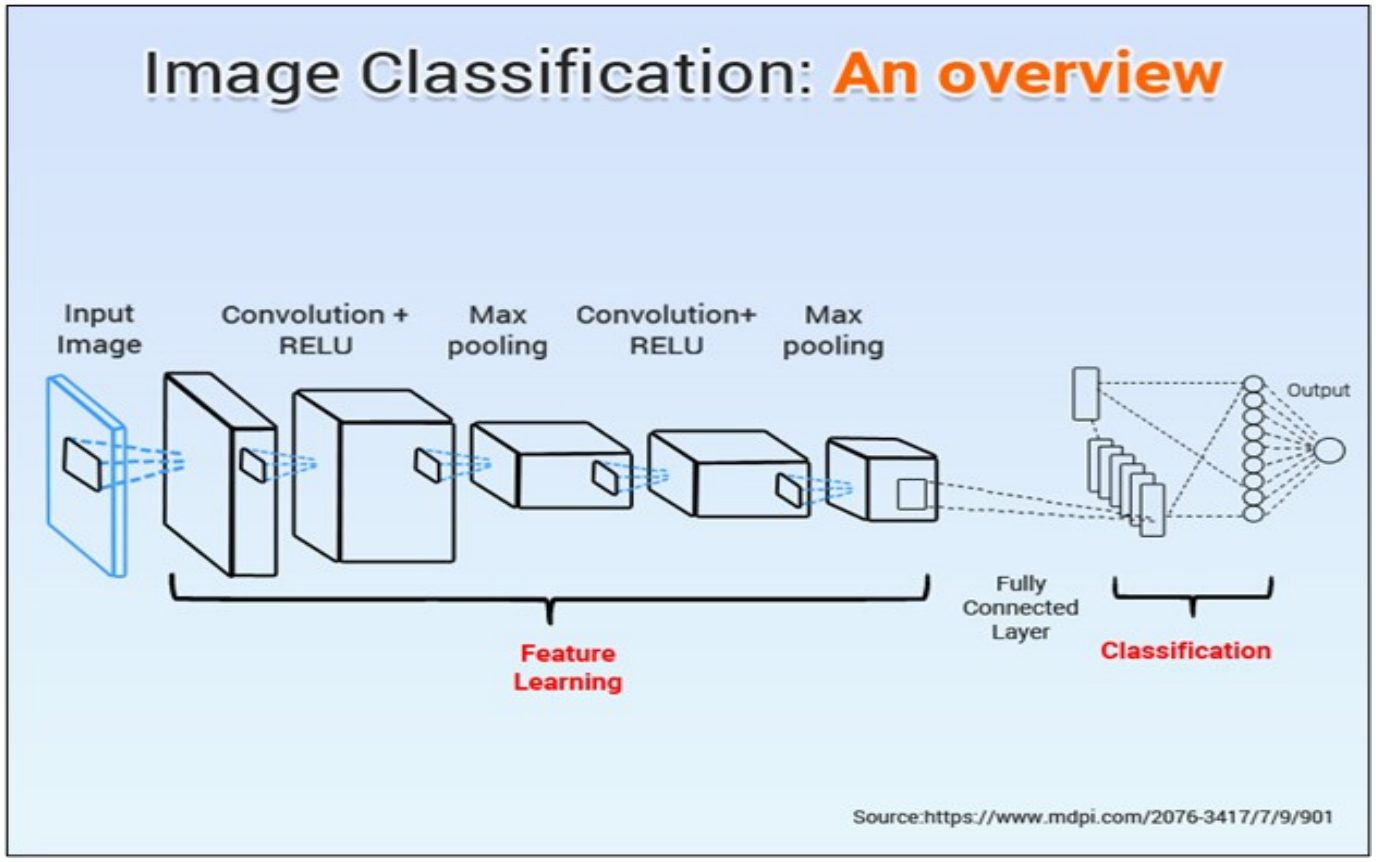


Figure 2.2. Overview of Image classification.[16]

2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, offering unparalleled accuracy in tasks such as image recognition, classification, and segmentation. CNNs have been applied to visual tasks since the late 1980s. However, despite a few scattered applications, they were dormant until the mid-2000s when the developments in computing power and the advent of large amount of labeled data, supplemented by improving algorithms, contributed to their advancement and brought them to the forefront of neural network renaissance that has seen rapid progression since 2012 [17]. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from image data. This architecture draws inspiration from the biological processes observed in the human visual cortex, employing layers with convolving filters that process data in a grid-like topology. Figure 2.3 describes a basic CNN architecture [18].

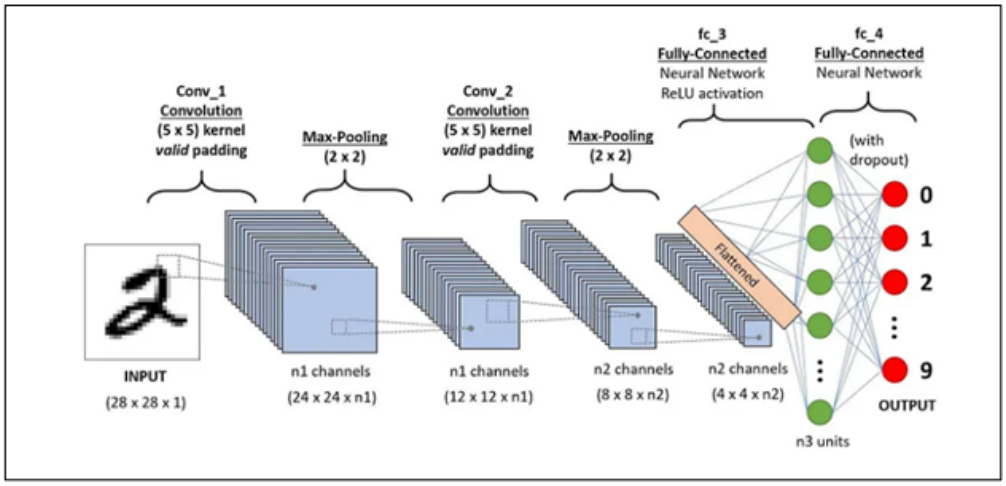


Figure 2.3. Basic CNN Architecture.[18]

2.4 Core Layers of CNNs

CNN architectures are composed of several types of layers, each with a unique role in processing the input data.

2.4.1 Convolutional Layer

The convolutional layer is the primary building block of a CNN. It applies a set of learnable filters to the input image to create feature maps. These filters are small in spatial dimension but extend through the full depth of the input volume. As the filter slides (or convolves) around the input image, it performs element-wise multiplication and produces a feature map that emphasizes features detected by the filter. This process allows the network to capture spatial hierarchies and patterns such as edges, shape and textures. To explain this operation let us consider an example of a 4 x 4 gray scale image with 2 x 2 random weights- initialized kernel. First, the kernel slides over the image horizontally and vertically. In addition, the dot product between the input image and the kernel is determined, where their corresponding values are multiplied and then summed up to create a single scalar value, calculated concurrently. This process is repeated until no further sliding is possible. The calculated dot product value represents the feature map of the output [19]. Figure 2.4 shows the primary calculation of each step of Convolutional layer [19].

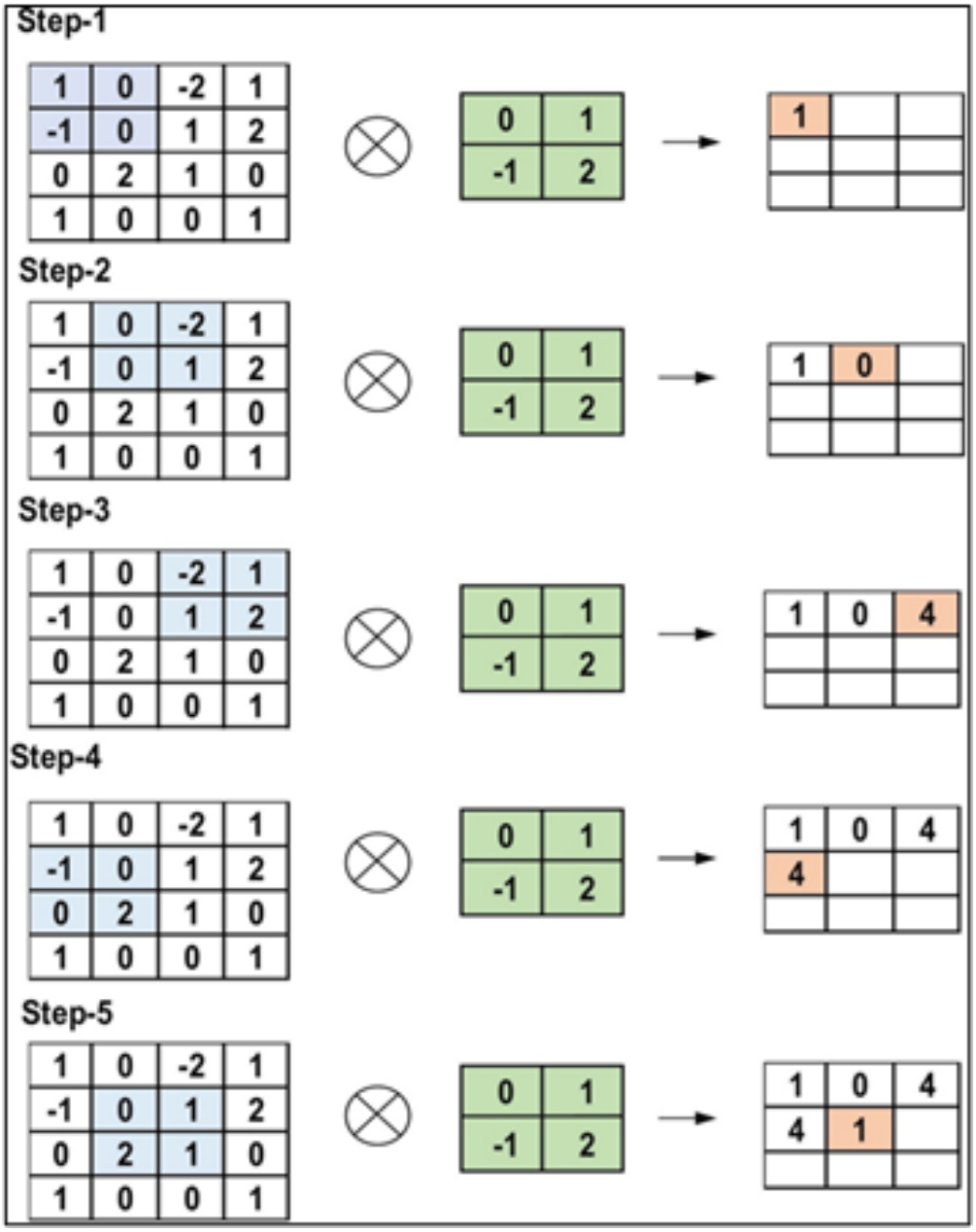


Figure 2.4. Calculation at each step of Convolutional Layer.[19]

2.4.2 Activation Layer (ReLU)

Activation layers introduce non-linearities into the output of neurons in a network. Without activation functions, even with multiple layers, the neural network would still behave as a linear operation, limiting its ability to learn complex patterns in the data. There are different types of activation functions that are used in this layer. The most commonly used functions are:

- ReLU (Rectified Linear Unit): The most widely used activation due to its simplicity and efficiency. It outputs the input directly if it is positive, else, it will output zero. It introduces non-linearity while maintaining computational simplicity.
- Sigmoid: Transforms the input into a range between 0 and 1. It's often used in the output layer of binary classification problems to interpret the output as a probability. However, it's less favored in hidden layers due to problems like vanishing gradients.
- Tanh (Hyperbolic Tangent): It is similar to the sigmoid but outputs values between -1 and 1. It is zero-centered, making it preferred in certain contexts over the sigmoid function.
- Softmax: Used primarily in the output layer of multiclass classification problems. It converts logits, the raw output scores from the last neural layer, into probabilities by taking the exponential of each output and then normalizing these values by dividing by the sum of all the exponentials.

2.4.3 Pooling Layer

Pooling (or subsampling) layers reduces the spatial dimensions (width and height) of the input volume for the next convolutional layer. This approach shrinks large-size feature maps to create smaller feature maps [19]. Pooling layer helps to make the representation approximately invariant to small translations of the input, reduce computation of upper layers, and provide a form of spatial compression that highlights the most salient features. This layer is useful to decrease the computational load, memory usage, and the number

of parameters, while extracting dominant features which are invariant to small changes, distortions, or translations in the input images. There are several types of pooling layers, but the three most common are:

- Max Pooling: This is the most frequently used form of pooling in CNN architectures. It splits the input image into a set of non-overlapping rectangles and outputs the maximum value of each sub-region. This has the effect of capturing the most prominent features in each region of the input, ensuring that these features are retained through subsequent layers of the network.

- Average Pooling: It calculates the average value of each region covered by the filter. This approach results in a smoother down sampled output, capturing the general presence of features in each region rather than focusing on the most prominent feature.

- Global Pooling: Instead of focusing on a specific region defined by the filter size, global pooling calculates the pool over the entire map to produce a single value. Global Max Pooling and Global Average Pooling are the two main types, often used before the final classification layer in a network.

There are different types of pooling operations utilized in pooling layers. These include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling. The most frequently used pooling methods are min, max and GAP. Figure 2.5 illustrates three types of pooling operations [19].

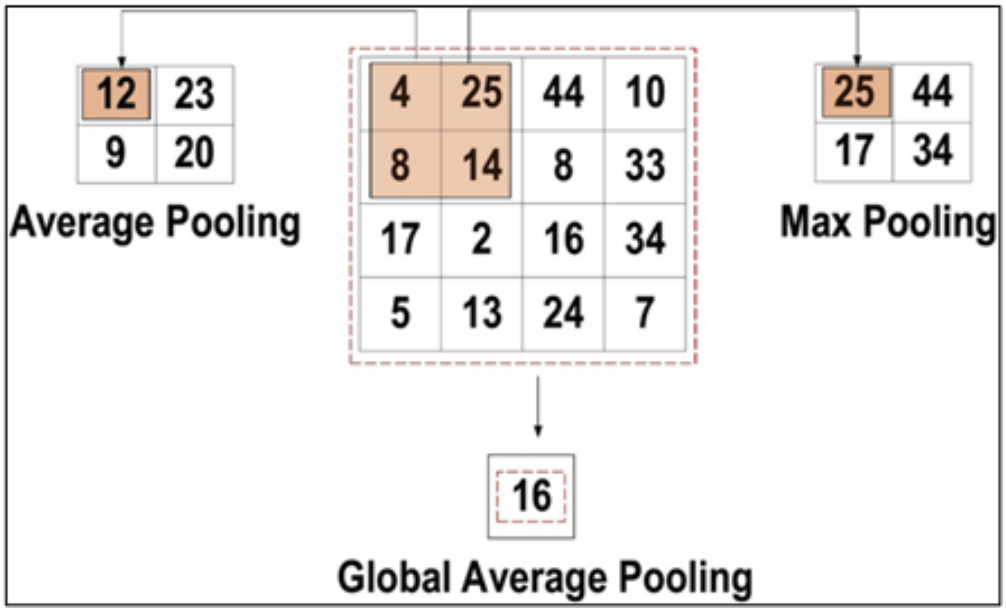


Figure 2.5. Three types of pooling operations.[19]

Average-pooling and max-pooling operations along the channel axis is utilized to obtain inter-spatial relationships of features and can be defined mathematically as shown in equation [19]:

$$y = \sigma(N_f(x)) \times x \tag{2.1}$$

2.4.4 Fully Connected Layer

Towards the end of the network, fully connected layers perform high-level reasoning by taking the outputs from the preceding layers and computing the class scores. In a fully connected layer, neurons have full connections to all activation in the previous layer. These layers are typically placed near the end of CNN architectures to perform classification based on the features extracted by the convolutional layers and condensed by the pooling layers [20]. Fully connected layers combine and transform the extracted features into a format suitable for making predictions. They integrate localized features learned by previous layers across the entire input domain. In classification tasks, the final fully connected layer usually has as many neurons as there are output classes, with a softmax activation function to output probabilities for each class. For regression tasks, the last fully connected layer might have a single neuron (for single-output regression) or multiple neurons for multi-output regression tasks, typically with linear activation. Using activation functions, fully connected layers can learn non-linear combinations of the high-level features extracted by the network, which is crucial for solving complex problems that cannot be addressed by linear models. Figure 2.6 illustrates fully connected layers [19].

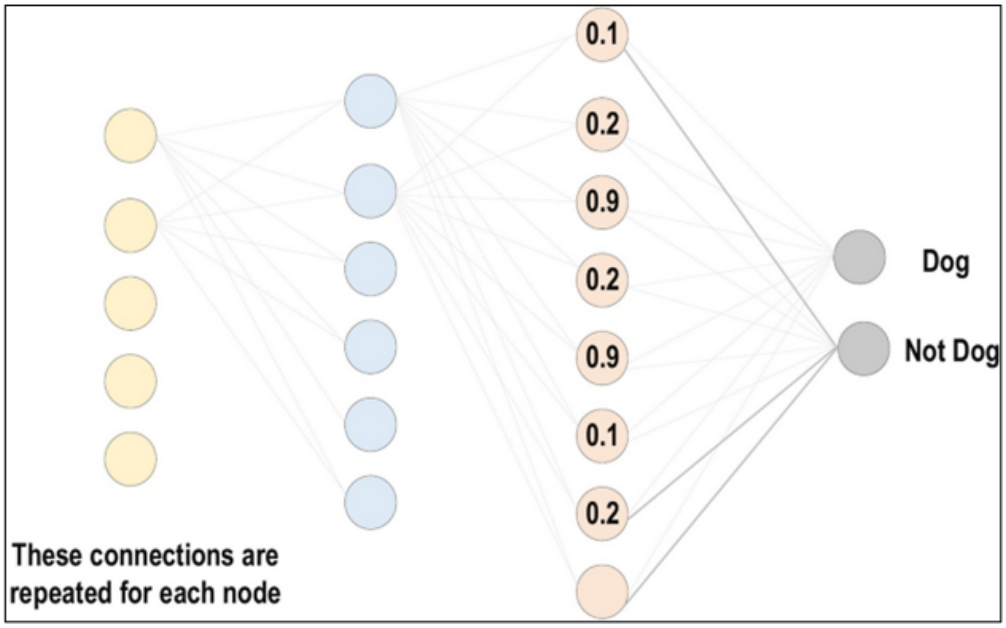


Figure 2.6. Fully Connected Layer.[19]

2.4.5 Classification Layer

The output classification layer, often the final layer in a neural network model designed for classification tasks, plays a crucial role in determining the predicted category for each input. This layer typically employs a softmax function in multi-class classification scenarios to convert the logits (raw prediction scores from the previous layer) into probabilities that sum up to one. In a multi-class classification problem, the output classification layer uses a softmax activation function. The softmax function exponentiates and normalizes the logits, transforming them into a probability distribution over predicted output classes. This makes it easier to interpret the network's output as confidence scores across the classes.

2.5 Regularization in CNN

Regularization is a crucial technique in the training of Convolutional Neural Networks (CNNs) to prevent overfitting. Overfitting occurs when a model learns the noise and random fluctuations in the training data to the extent that it performs poorly on new, unseen data. Regularization techniques are employed to simplify the model, making it less likely to capture the noise in the training data, thus improving its generalization capabilities to new data. There are several common regularization techniques used in CNNs.

2.5.1 L1 and L2 Regularization

- L1 Regularization: Adds a penalty equal to the absolute value of the magnitude of coefficients. It can lead to the elimination of some features' influence on the model by making their coefficients zero. It is useful for feature selection.
- L2 Regularization: Adds a penalty equal to the square of the magnitude of coefficients. This discourages large coefficients but does not set them to zero. It's useful for models that benefit from all features being included.

Both L1 and L2 regularization are applied to the loss function, adding a term that increases with coefficient size, thus discouraging large weights.

2.5.2 Dropout

Dropout is a powerful regularization technique in deep learning. During training, some number of layer outputs are randomly ignored or "dropped out." This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. As a result, the network becomes less sensitive to the specific weights of neurons. This leads to each neuron becoming more independently useful as they cannot rely on the presence of other neurons. It effectively makes the network more robust and reduces overfitting.

2.5.3 Data Augmentation

Data augmentation artificially increases the size of the training dataset by applying various transformations to the input images. These transformations include rotation, scaling, cropping, flipping, and altering the color settings. By training the network on this augmented dataset, the model can learn more generalized features, making it robust against overfitting and improving its performance on unseen data.

2.5.4 Batch Normalization

Batch Normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing it by the batch standard deviation. This process stabilizes the learning process and dramatically reduces the number of training epochs required to train deep networks. While its primary purpose is to make the optimization landscape smoother, BN also has a regularizing effect, as it adds some noise to the layers' activations, like dropout.

2.5.5 Early Stopping

Early stopping involves halting the training process when the performance on a validation set starts to deteriorate or ceases to improve substantially. By monitoring the validation loss or validation accuracy during training, training can be stopped at the point when per-

formance on the validation set is maximized. This prevents overfitting by avoiding the point where the model starts to learn noise in the training data.

2.6 Overfitting and Underfitting in CNN

Overfitting

Overfitting is a common challenge in training convolutional neural networks (CNNs), as well as in machine learning models at large. It occurs when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. Essentially, an overfitted model is too complex, capturing spurious correlations in the training data that do not generalize to unseen data. Causes of overfitting are:

- Too Complex Model: Having too many parameters compared to the number of observations can make the model fit not just the underlying pattern but also the noise in the dataset.
- Insufficient training data: A small dataset increases the likelihood of the model learning noise in the data as significant patterns.
- Noisy or Unrepresentative Training Data: Data that doesn't well represent the underlying problem, or that contains a lot of noise, can lead the model to learn irrelevant patterns.

Underfitting

Underfitting is the opposite problem of overfitting in machine learning, including in the context of convolutional neural networks (CNNs). It occurs when a model is too simple to capture the underlying structure of the data, leading to poor performance on both the training data and unseen data. Essentially, an underfitted model has not learned the relevant patterns in the training data well enough to make accurate predictions. Figure 2.7 illustrates Overfitting and Underfitting issues [19]. Causes of underfitting are:

- Too Simple Model: A model that is too simplistic might not have enough parameters or layers to capture the complexity of the data's underlying structure.
- Insufficient Training: Underfitting can also occur if the model has not been trained for enough epochs, i.e., it hasn't had the chance to learn from the training data adequately.
- Poor Feature Selection: If the input features to the model do not contain enough information or are not relevant to the task at hand, the model might fail to make accurate predictions.

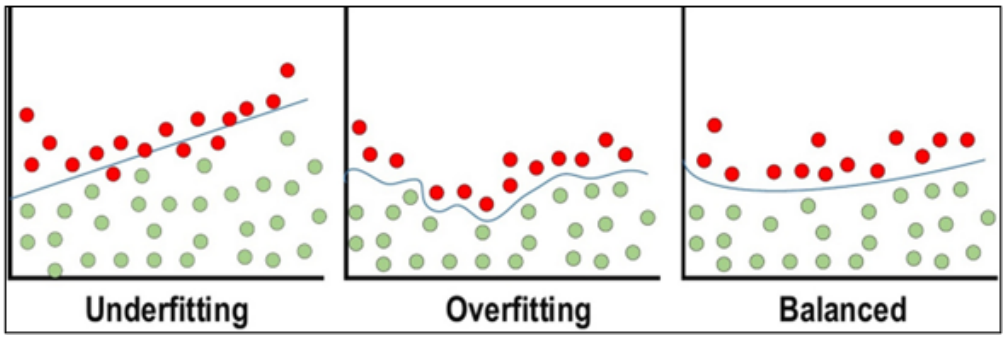


Figure 2.7. Overfitting and Underfitting issues.[19]

2.7 Gradient Descent:

Gradient descent is a fundamental optimization algorithm used to minimize the loss function in machine learning and deep learning models. It's crucial for training models effectively, allowing them to learn from the data by iteratively adjusting the model's parameters (weights and biases) to reduce errors in predictions. Gradient descent operates on the principle that if you want to minimize a function, you should move in the direction opposite to the gradient (the slope) of the function at that point. The gradient provides the direction of the steepest ascent. Adjust the parameters in the direction opposite to the gradient to reduce the loss. This is done using the equation:

$$\Theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta J(\theta)$$

where Θ_{new} represents the new parameters,

θ_{old} represents the old parameters,

η is the learning rate (a small, positive value determining the step size), and

$\nabla_\theta J(\theta)$ is the gradient of the loss function J with respect to θ.

(2.2)

2.8 Evolution of EfficientNet Architecture

EfficientNet architectures, introduced by Tan and Le (2019) [21], mark a significant advancement in the design of convolutional neural network. The motivation behind EfficientNet was the observation that most available CNNs were scaled in a rather arbitrary manner - either increasing the depth (number of layers), width (number of channels), or image resolution to improve accuracy, often leading to a significant increase in computational cost without proportional gains in performance. EfficientNet introduced a novel scaling method that uses a compound coefficient to uniformly scale network width, depth, and resolution in a principled manner. This method, called compound scaling, is grounded in the observation that the different dimensions of a network (depth, width, and resolution) are not independent but rather complementary, and scaling them together yields better efficiency and performance. Figure 2.8 illustrates EfficientNet Architecture [22].

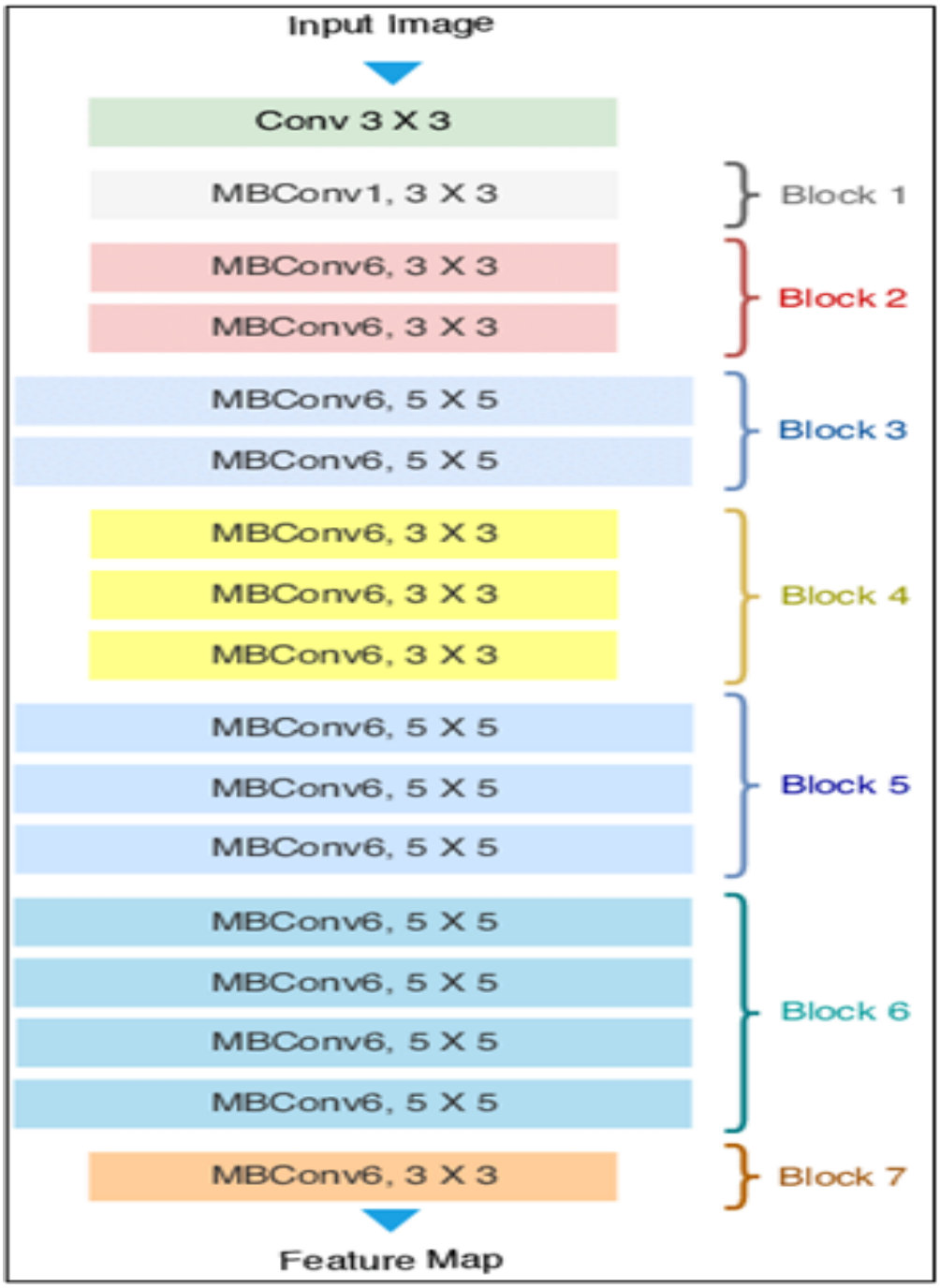


Figure 2.8. EfficientNet Architecture.[22]

2.9 CIFAR-10 as a Benchmark Dataset

The CIFAR-10 dataset, introduced by Krizhevsky et al., has emerged as a critical benchmark for assessing the performance of image classification models. Comprising 60,000 32x32 color images across ten classes, CIFAR-10 challenges models to accurately recognize and classify diverse objects under various settings. Its widespread use in the deep learning community has facilitated numerous advancements in model accuracy and efficiency, serving as a testament to the progress in computational vision technologies.

2.10 Transfer Learning and Fine Tuning

Transfer learning and fine-tuning are powerful techniques in the field of machine learning and deep learning that leverage the knowledge gained from one problem to solve similar ones. These approaches are particularly useful in situations where the available labeled data for a specific task is limited.

- Transfer Learning: Transfer learning involves taking a pre-trained model (a model trained on a large dataset) and applying it to a different but related problem. The intuition behind transfer learning is that if a model has learned certain features from a large dataset, these features could be relevant and useful for a new task.
- Fine Tuning: Fine-tuning is a specific case of transfer learning that involves minor adjustments to the pre-trained model to adapt it to the new task. Unlike using the model solely as a feature extractor, fine-tuning allows for the modification of the pre-existing model weights.

Advantages of Transfer learning and Fine tuning:

- Efficiency: These techniques significantly reduce the computational cost and time required to develop deep learning models by reusing existing architectures and weights.
- Improved Performance: Models pre-trained on large datasets have been exposed to a wider variety of features, which can lead to better generalization and performance on similar but smaller datasets.

- Flexibility: Transfer learning and fine-tuning can be applied across different domains, such as computer vision, natural language processing, and more, making it a versatile approach to solving machine learning problems.

2.11 Model Interpretability and Evaluation

As deep learning models, particularly CNNs, become more complex, the importance of model interpretability grows. Researchers have developed various techniques to visualize and understand how these models make decisions, including feature map visualizations and saliency maps. Furthermore, evaluating the performance of these models involves a comprehensive set of metrics, including accuracy, precision, recall, and F1 scores, alongside confusion matrices and classification reports. These evaluation tools provide crucial insights into the models' strengths and weaknesses, guiding further improvements and adaptations.

3. BASELINE ARCHITECTURE

The EfficientNet architecture, introduced by Mingxing Tan and Quoc V.Le in 2019 [23], represents a significant advancement in scaling up Convolutional Neural Networks (CNNs) for improved performance and efficiency. The key innovation behind EfficientNet is the systematic and compound scaling of network width, depth, and resolution, guided by a set of fixed scaling coefficients. EfficientNet represents a milestone in CNN architecture design, demonstrating that carefully balanced scaling of network dimensions can lead to significant improvements in model performance and efficiency. Its principled approach to scaling, rooted in the compound coefficient, sets it apart from previous models and offers a versatile architecture for a wide range of image recognition tasks. The EfficientNet model architecture, starting with the EfficientNetB0 baseline, provides a solid foundation for advancements in deep learning, enabling researchers and practitioners to achieve state-of-the-art results in computer vision challenges. This section outlines the EfficientNet model architecture and its foundational principles, offering a comprehensive overview of the baseline architecture.

3.1 Introduction to EfficientNet

EfficientNet is based on the insight that the scaling of network dimensions (depth, width, and input resolution) needs to be balanced to achieve optimal performance. Prior approaches often scaled these dimensions arbitrarily, leading to suboptimal performance or inefficient use of computational resources. EfficientNet introduces a principal method of scaling, using a compound coefficient (ϕ) to uniformly scale network width, depth, and resolution in a coherent way that maximizes the efficiency and effectiveness of the model.

3.2 EfficientNet Models Series

The EfficientNet series consists of B0 to B7 models, each offering a trade-off between accuracy and computational efficiency. As the model number increases, so do the depth, width, and resolution, according to the compound scaling rule. This allows for flexible deployment of EfficientNet models across a wide range of computational environments and

applications, from mobile devices to powerful cloud servers. Figure 3.1 shows the impact of number of parameters on the model accuracy.[21]

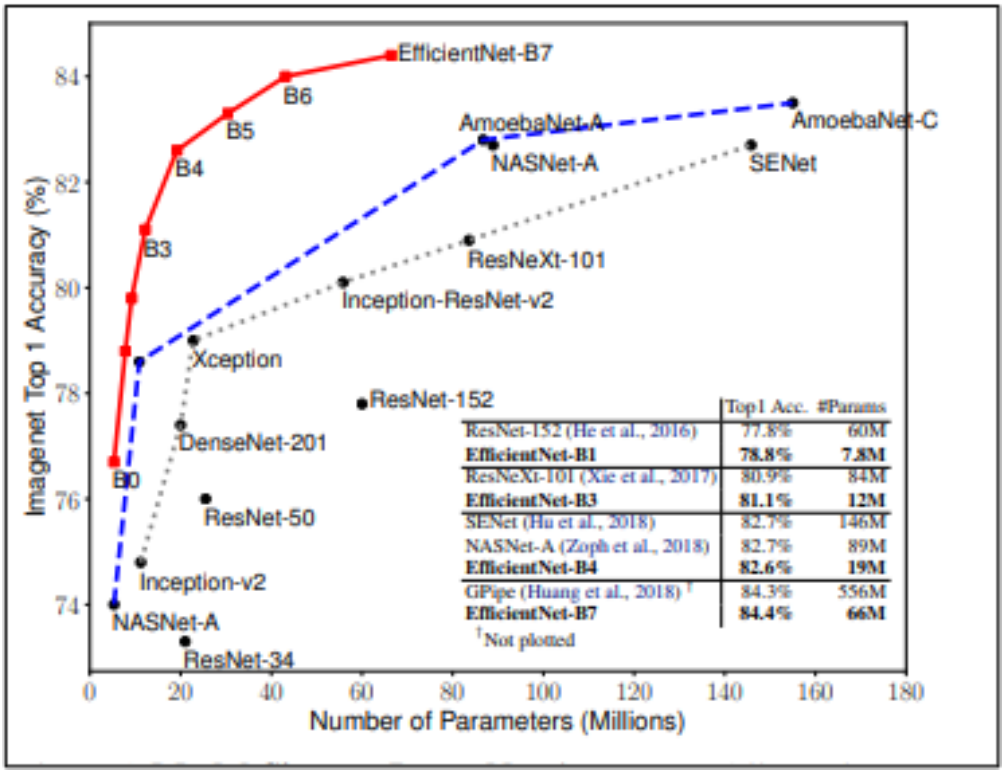


Figure 3.1. Model size Vs ImageNet Accuracy.[21]

3.3 Baseline Architecture: EfficientNetB0

3.3.1 EfficientNetB0 Core

At the heart of the baseline architecture is EfficientNetB0, a convolutional neural network that represents a significant advancement in efficient and scalable model design. EfficientNetB0 was initially trained on the ImageNet dataset, which comprises over a million images across 1000 classes, providing a robust foundation for feature extraction. EfficientNet leverages NAS to search for the baseline EfficientNetB0 that has better trade off on accuracy and FLOPs [24]. The core of EfficientNetB0 incorporates MBConv blocks (Mobile Inverted Bottleneck Convolution), which are optimized for mobile and edge devices, offering an excellent balance between latency and accuracy. These blocks utilize depthwise separable convolutions

that significantly reduce the number of parameters while maintaining model effectiveness. To tailor the pre-trained EfficientNetB0 for CIFAR-10 classification, the model is adapted as follows:

- Input Resizing: The input images are resized to 224x224 pixels to match the input size expected by EfficientNetB0, facilitating the use of pre-trained weights without architectural modifications.
- Global Average Pooling 2D (GAP): Post convolutional processing, a Global Average Pooling layer is included to reduce each feature map's spatial dimensions to a single value. This operation condenses the model's extracted features into a form suitable for classification, reducing the total number of parameters and computational cost.
- Output Layer: The final layer of the model is a Dense layer with 10 units, corresponding to the 10 classes of CIFAR-10, and utilizes a softmax activation function. This layer computes the probability distribution across the 10 classes, facilitating the classification task. Figure 3.2 illustrates the Baseline model architecture.

3.3.2 Mobile Inverted Bottleneck Convolution (MBConv)

The Mobile Inverted Bottleneck Convolution (MBConv) is a key architectural component within EfficientNet and other modern convolutional neural networks, originally popularized by MobileNetV2. It is an evolution of depthwise separable convolutions, designed to provide an efficient mechanism for building lightweight and powerful deep learning models. MBConv blocks are particularly effective in mobile and edge computing scenarios, where computational resources and power are limited. Components of MBConv:

- Inverted Residual Structure: Unlike traditional convolutional blocks that first reduce the input channel dimension before applying convolution, MBConv blocks invert this process. They start by expanding the input's channel dimension using a 1x1 convolution (pointwise convolution), apply depthwise convolution for spatial feature extraction, and then project the features back to a lower dimension with another 1x1 convolution [25]. This expansion and projection process helps in extracting richer features

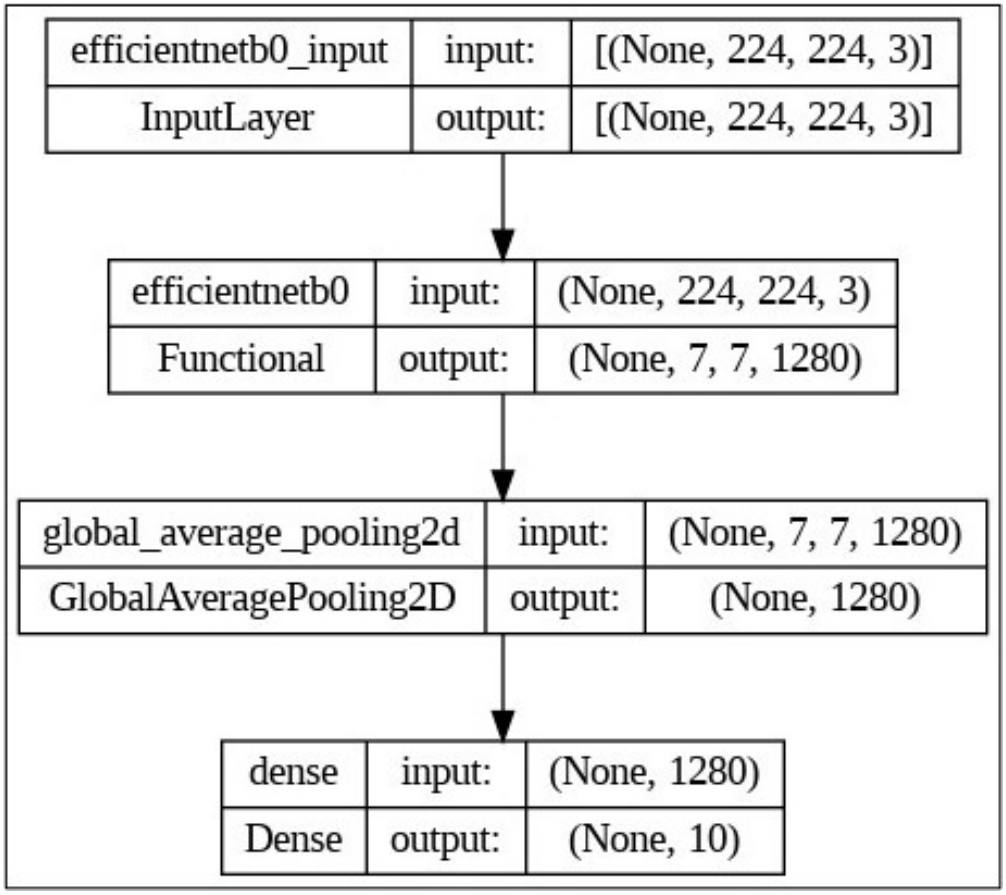


Figure 3.2. Baseline Architecture.

with fewer parameters. Inspired by MobileNetV2 and Inverted Residuals, MBConv is the primary building block of EfficientNetB0, utilizing depthwise separable convolutions with an expansion factor and squeeze-and-excitation optimization. This block significantly enhances the model's efficiency by reducing the number of parameters and computational cost while maintaining performance.

- Depthwise Separable Convolution: Central to MBConv is the use of depthwise separable convolution, which splits a standard convolution into two separate layers: depthwise convolution and pointwise convolution. The basic idea of a Depthwise convolution is to replace a full convolutional operator with a factorized version that splits convolution into two separate layers [26]. It applies a single filter per input channel (reducing

computational cost), and pointwise convolution (1x1 convolution) combines the output channels from the depthwise convolution. This separation significantly reduces the number of computations and parameters compared to standard convolutions.

- Squeeze and Excitation (SE) Optimization: Integrated within the MBConv block, the squeeze-and-excitation mechanism adaptively recalibrates channel-wise feature responses by emphasizing important features and suppressing less useful ones. This is achieved by squeezing global spatial information into a channel descriptor, using fully connected layers to capture channel-wise dependencies, and then rescaling the original feature maps by these learned importance weights.

Advantages of MBConv:

- Efficiency: MBConv reduces the number of parameters and computational cost without significantly compromising the model's performance, making it ideal for applications where computational resources are limited.
- Flexibility: The modular nature of MBConv blocks allows them to be easily integrated into various network architectures, enabling the construction of models tailored to specific performance and efficiency requirements.
- Performance: Despite its efficiency, the MBConv block, especially when combined with squeeze-and-excitation optimization, provides significant gains in accuracy by focusing on the most relevant features within an image.

3.3.3 Compound Scaling Method

Compound scaling is a systematic approach to scaling the dimensions of a neural network to improve its performance and efficiency. Introduced with the EfficientNet architecture, compound scaling simultaneously scales the network's depth (number of layers), width (number of channels), and resolution (size of the input images) with a fixed set of scaling coefficients. This approach is based on the insight that simply increasing the size of a network (e.g., making it deeper or wider) does not always lead to proportional gains in performance

due to the complex interplay between these dimensions. EfficientNet starts with a baseline architecture (EfficientNetB0) designed for optimal performance on a specific resource constraint. Compound scaling is then applied to scale up this baseline model to create a family of models (EfficientNetB1 to EfficientNetB7) that offer a spectrum of trade-offs between accuracy and computational cost. As increases, the network becomes deeper (more layers), wider (more channels), and higher in resolution (larger input images), which generally leads to higher accuracy but requires more computational resources.

Advantages of Compound scaling:

- Efficiency: By scaling all dimensions of the network in a balanced manner, compound scaling ensures that additional computational resources are used effectively, leading to better performance without wasteful use of resources.
- Flexibility: The approach provides a systematic way to create a family of models with different sizes, making it easier to choose a model that fits the constraints of a specific application, whether it be low-latency mobile apps or high-accuracy cloud applications.
- Improved Performance: Compound scaling has been shown to achieve state-of-the-art performance on benchmark datasets like ImageNet, demonstrating its effectiveness in building high-performing and efficient deep learning models.

3.3.4 Scaling EfficientNet

To create models B1 through B7, EfficientNet employs a simple yet effective compound scaling method, using a compound coefficient (ϕ) to govern the scaling of network depth, width, and resolution. The scaling coefficients are determined based on a grid search that optimizes the balance between accuracy and efficiency on a target dataset, such as ImageNet. As ϕ increases, the network becomes deeper, wider, and higher in resolution, leading to higher accuracy but also increased computational complexity. Figure 3.3 explains Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio [21].

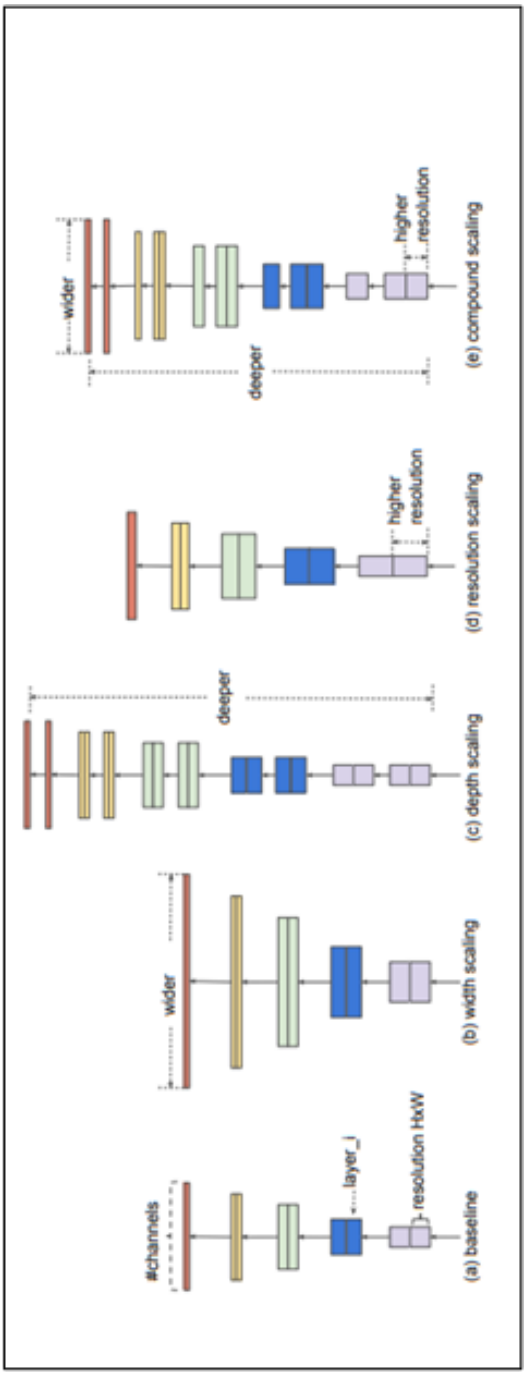


Figure 3.3. Model Scaling. [21]

4. PROPOSED MODEL ARCHITECTURE

The proposed model represents a significant advancement in applying deep learning techniques to the task of image classification, specifically tailored for the CIFAR-10 dataset. By integrating the EfficientNetB0 architecture pre-trained on ImageNet with additional customization, the model achieves superior performance while maintaining computational efficiency. This section delves into the architecture, training regimen, and evaluation strategies of the proposed model.

4.1 Architecture Overview

The core of the proposed model is the EfficientNetB0 architecture, renowned for its balanced scaling of network depth, width, and resolution. The EfficientNetB0 base model is employed as a feature extractor, with the following modifications to cater to the CIFAR-10 dataset.

- Input Adaptation: Given the CIFAR-10 images are small (32x32 pixels), they are resized to 224x224 pixels to match the input dimension expected by EfficientNetB0. This resizing is crucial for leveraging the pre-trained weights effectively, allowing the model to benefit from the features learned on the more extensive and diverse ImageNet dataset.

- Global Average Pooling (GAP) Layer: Following the base model, a GAP layer is added to condense the extracted features into a form that is more manageable for classification. This layer reduces the spatial dimensions of the feature maps to a single vector per feature map, significantly decreasing the number of parameters and thus the risk of overfitting.

- Dense Layer: A dense layer, also known as fully connected layer, is a classic type of neural network layer where each input is connected to each output by a weight. It is a feedforward layer that is deeply connected, which means that every neuron in the Dense layer receives input from the previous layer.

- Neurons: Each neuron in a Dense layer performs a weighted sum of the inputs, adds a bias term, and passes the result through an activation function.
- Weights and Biases: Weights and biases are the parameters that the layer learns during the training process. Each neuron has a weight for each input and single bias term.
- Activation Function: It is used to introduce non-linearity into the output of a neuron. Common activation functions include ReLU, sigmoid and tanh.

The proposed model uses a Dense layer with 512 units and ReLU activation function that introduces the model to learn complex patterns from the condensed features. The ReLU activation function is a linear activation function which has the thresholding at zero as shown in Equation 4.1. The convergence of gradient decent can be accelerated by applying ReLU [19].

$$\text{rect}(x) = \max(0, x) \tag{4.1}$$

The GAP layer condenses spatial information into a single vector per channel, preserving essential features while significantly reducing data dimensionality. Following this layer with a Dense layer with 512units allows for a rich, comprehensive interpretation of these features. Each unit in the dense layer can learn to recognize complex patterns and relationships among the features extracted by the previous layer, improving the models ability to make accurate predictions. The dense layer introduces additional parameters (weights and biases) that increase the models capacity to learn. With 512 units the model gains substantial boost in learning complexity, which is essential for tackling complicated tasks like distinguishing between the numerous classes in CIFAR datasets. The key here is that this complexity doesnt lead to overfitting. The Global Average Pooling ensures that the model focuses on global features by averaging out the spatial information. The dense layer that follows can learn from these global features more effectively, as each unit can contribute to the final decision-making process based on a comprehensive, distilled set of information about the entire image. Having a dense layer with a significant number of units like 512 offers

flexibility in model design. It allows for adjustments based on the complexity of the task or the specificity of the data being modeled. This scalability is vital for fine-tuning the model to achieve optimal performance on a variety of tasks, from simple to highly intricate image classification challenges.

- Dropout Layer: Dropout layer is a regularization technique used to prevent overfitting in neural networks. It randomly drops a proportion of the neurons (and their connections) during the training phase, forcing the network to learn redundant representations and thus improves generalization. Dropout is a common technique used in computer vision and we apply it to the output after the global average pooling occurs in the final layer [27]. Characteristics of Dropout layer are:
  - Reduction of Co-adaptation: By dropping different sets of neurons, it prevents neurons from co-adaptation.
  - Improving Generalization: Since each mini batch is trained on a different thinned network, the model generalizes better to unseen data.
  - Simple Yet Effective: Despite its simplicity, dropout is a powerful tool in deep learning to improve model robustness.

The proposed model uses a Dropout layer with a rate of 0.5, which is employed to further mitigate overfitting by randomly "dropping" a portion of the neurons, forcing the model to learn more robust features. Dropout is a regularization technique designed to perform overfitting, a common problem where the model performs well on the training data but poorly on unseen data. By randomly setting the output to 50 percent of the neurons in the dense layer to zero during training, dropout forces the network to learn more robust features that are not reliant on any small set of neurons. This improves the models generalization ability to new, unseen data. Since dropout randomly drops units during the training, the network cannot depend on any single unit to represent a feature. This compels the network to learn redundant representations for the data, ensuring that important features are captured by multiple neurons. This redundancy makes the model more fault tolerant and enhances the ability to recognize patterns even if some data points are missing or noisy. Dropout can be

interpreted as a way of performing the model averaging with neural networks. Each training iteration with dropout uses a different thinned model, and the final model can be seen as an average of these smaller models. Model averaging is a well-known technique to improve models performance by reducing variance without substantially increasing bias.

- Classification Layer: The classification layer serves as the decision-making component of the neural network, translating the features learned by the network during training into predictions for given input. This Dense layer with 10 neurons or units are also known as fully connected layer. Each neuron corresponds to one of the 10 classes in the CIFAR-10 dataset- airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The activation function used in this layer is softmax function. Unlike other activation functions like ReLU that act on individual neurons independently, the softmax function considers the outputs of all neurons in the layer. It assigns probability to each class in a way that the sum of probabilities of all classes equals to one. This function is particularly suitable for multi-class classification tasks, as it allows the model to give probabilistic interpretation of each class being the correct one for the input. The output of the softmax function is a probability distribution that indicates the likelihood of the input image belonging to each of the 10 classes. The class with the highest probability is typically considered the models prediction. The role of this layer is critical as it encapsulates the essence of the learned patterns and features from previous layers and maps them into a space where the classification decision is made. It essentially forms the bridge between the feature extraction and decision-making process. During the training phase, the networks weight and biases are adjusted to minimize the difference between the predicted probability distribution and the true distribution (where the class has a probability of 1 and the rest 0). This process involves calculating the loss using a suitable loss function, such as categorical cross-entropy, and using an optimization algorithm to update the parameters. Figure 4.1 illustrates the proposed model architecture.

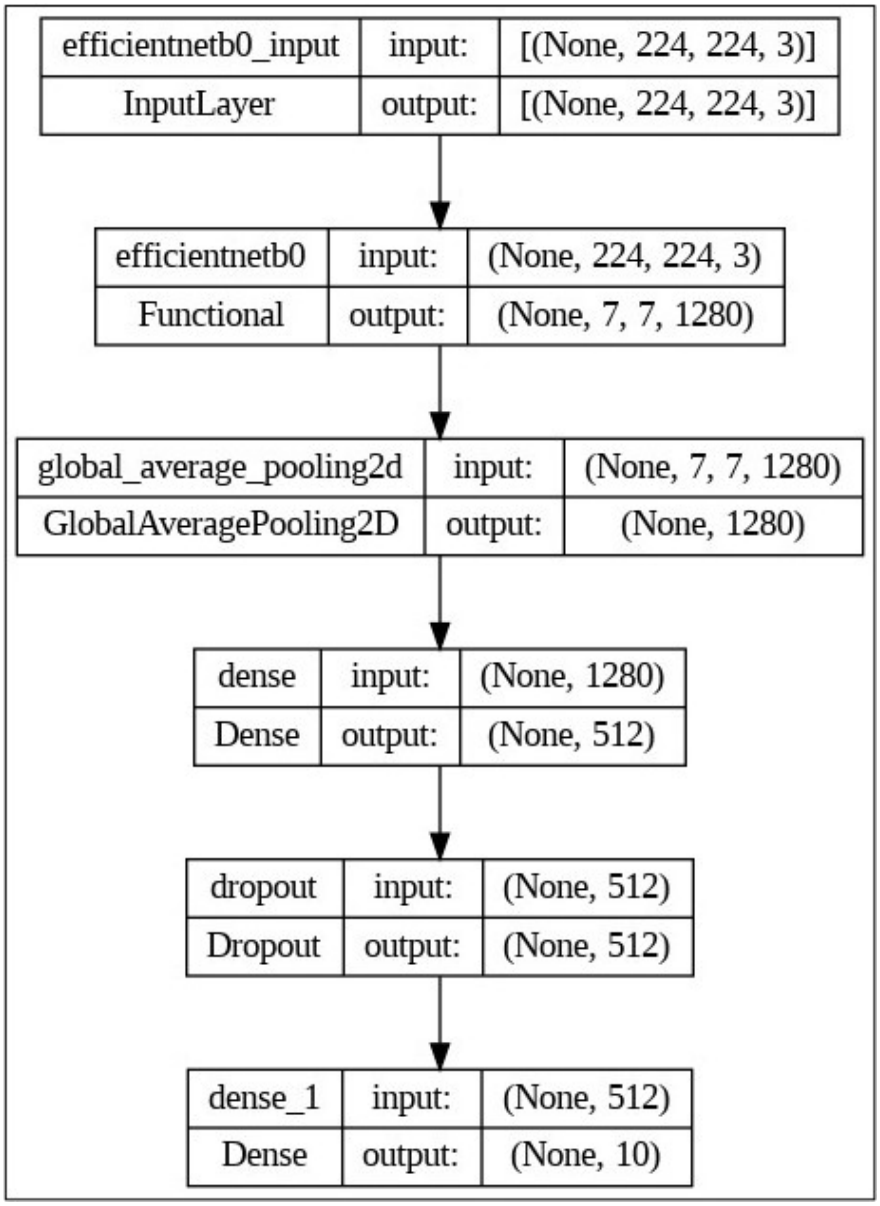


Figure 4.1. Proposed model architecture.

4.2 Training Regimen

4.2.1 Adam Optimizer

The Adam optimizer is a popular choice for training deep learning models due to its adaptive learning rate capabilities. Unlike traditional stochastic gradient descent, Adam adjusts the learning rate for each parameter individually, based on estimates of the first (mean) and second (uncentered variance) moments of the gradients. This makes it particularly effective for complex models and datasets with varied features, as it can navigate the optimization landscape more intelligently, leading to faster convergence.

4.2.2 Categorical Crossentropy Loss Function

This loss function is ideal for multi-class classification problems where each class is mutually exclusive. It measures the dissimilarity between the true distribution (the labels) and the predicted distribution, with the goal of minimizing this difference. In the context of CIFAR-10, where an image can belong to one of ten distinct classes, categorical crossentropy effectively guides the model to improve its predictions over time.

4.2.3 Early Stopping

To prevent overfittinga scenario where the model performs well on training data but poorly on unseen dataearly stopping monitors the validation loss (the models performance on a validation set) and stops the training process if the loss does not improve for a specified number of epochs. This strategy ensures that the model remains generalizable to new, unseen data by halting the training at the point where performance on the validation set starts to degrade.

4.2.4 ReduceLRPlateau

This callback reduces the learning rate when a metric has stopped improving, specifically focusing on the validation loss. By lowering the learning rate in small steps, the model can explore the optimization landscape more finely, potentially escaping local minima or making

more precise adjustments to the weights. This strategy is crucial for fine-tuning the model to achieve better convergence.

4.2.5 Training Phases

- Initial Training Phase: In this phase, the EfficientNetB0 base model weights are frozen, meaning they are not updated during training. The focus is on training the layers added on top of the base model, such as the global average pooling, dense, and dropout layers. This approach allows these newly added layers to learn from the high-level features extracted by the pre-trained EfficientNetB0 without disturbing the learned representations.

- Fine-Tuning Phase: After the initial training phase, a portion of the base model's layers is unfrozen, allowing them to be updated during training. This phase is referred to as fine-tuning. It enables the model to adjust the more abstract features extracted by the EfficientNetB0 layers to better align with the specific characteristics of the CIFAR-10 dataset. Fine-tuning is a powerful strategy for leveraging pre-trained models on new tasks, as it fine-tunes the pre-existing learned features to the nuances of the new dataset, often leading to significant improvements in model performance.

4.3 Performance Evaluation and Model Interpretation

The model's performance is meticulously evaluated using a validation set, with accuracy as the primary metric. Further insights are gained through the generation of confusion matrices and classification reports, offering a detailed view of the model's predictive capabilities across the various classes. The training and validation processes are visually analyzed by plotting the learning rate over time, along with training and validation accuracy and loss. These plots provide valuable insights into the model's learning dynamics, helping identify overfitting and underfitting patterns.

4.3.1 Confusion Matrix

A confusion matrix is a table often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows easy identification of confusion between classes, i.e., where the model is mislabeling the classes. The matrix is NŒN, where N is the number of classes being predicted. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class.

- True Positives (TP): Correctly predicted positive cases.
- True Negatives (TN): Correctly predicted negative cases.
- False Positives (FP): Incorrectly predicted positive cases.
- False Negatives (FN): Incorrectly predicted negative cases.

Metrics derived from Confusion matrix are:

- Accuracy: Overall, how often is the classifier correct? Accuracy = (TP + TN)/ (TP + TN + FP + FN)
- Precision: When it predicts positive, how often is it correct? Precision = TP/ (TP + FP)
- Recall: How often it predicts positive when the case is true positive? Recall = TP/ (TP + FN)
- F1 Score: A weighted average of precision and recall F1 Score = 2 x (Precision x Recall/ Precision + Recall)
- Specificity: How often it predicts negative when the case is true negative. Specificity: TN/ (TN + FP)

5. EXPERIMENTS AND RESULTS

The proposed model was initially evaluated on CIFAR-10 dataset. Additionally, to prove its robustness, the model was also evaluated on CIFAR-100 dataset. Both tests were ran using the below-mentioned training infrastructure. The results from the tests are also documented in this section.

5.1 Training Infrastructure

- GPU A100
- Python 3.10.12
- TensorFlow 2.15.0

5.2 Training and Testing Results

The proposed model was meticulously engineered and implemented using TensorFlow, a comprehensive, open-source platform that facilitates machine learning and deep learning model development. Through the utilization of TensorFlow, the model benefited from advanced features such as automatic differentiation, which greatly simplified the computation of gradients for optimization algorithms.

5.2.1 CIFAR-10 Dataset

CIFAR stands for Canadian Institute for Advanced Research and the number 10 signifies that this dataset consists of the following ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. This dataset consists of 60000 (32x32) color images in total, with 6000 images per class. Out of the 60000 images, 50000 are used for training the model while the remaining are retained for testing the model [28]. Figure 5.1 provides an overview of all classes in this dataset.

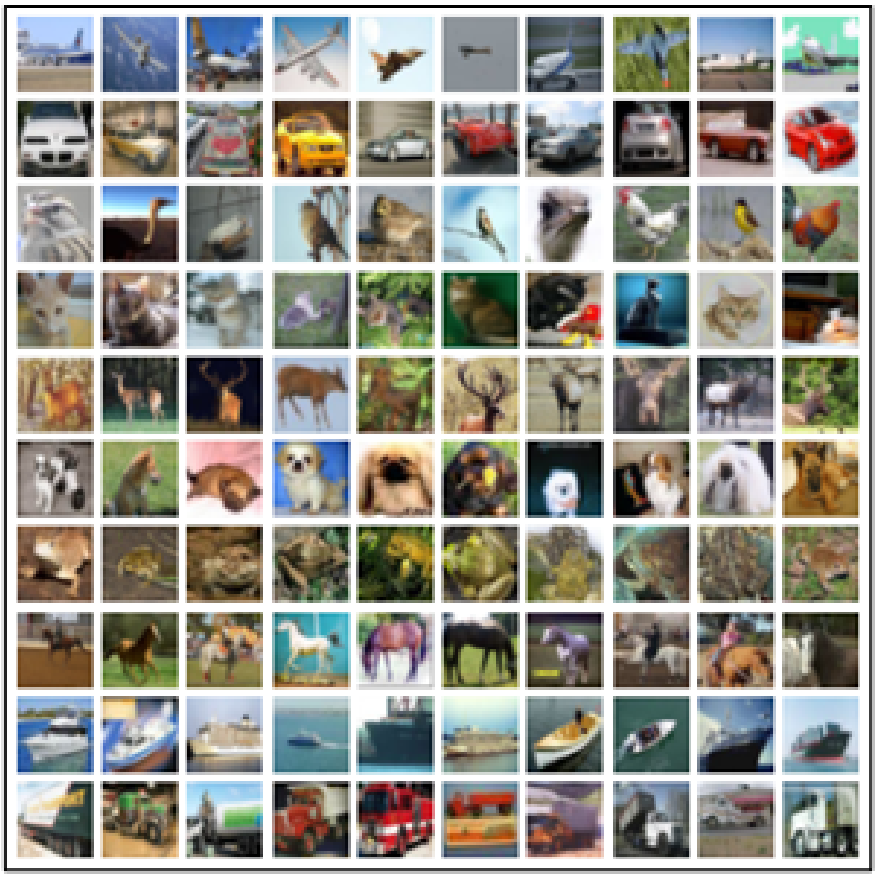


Figure 5.1. Image classes in CIFAR-10.

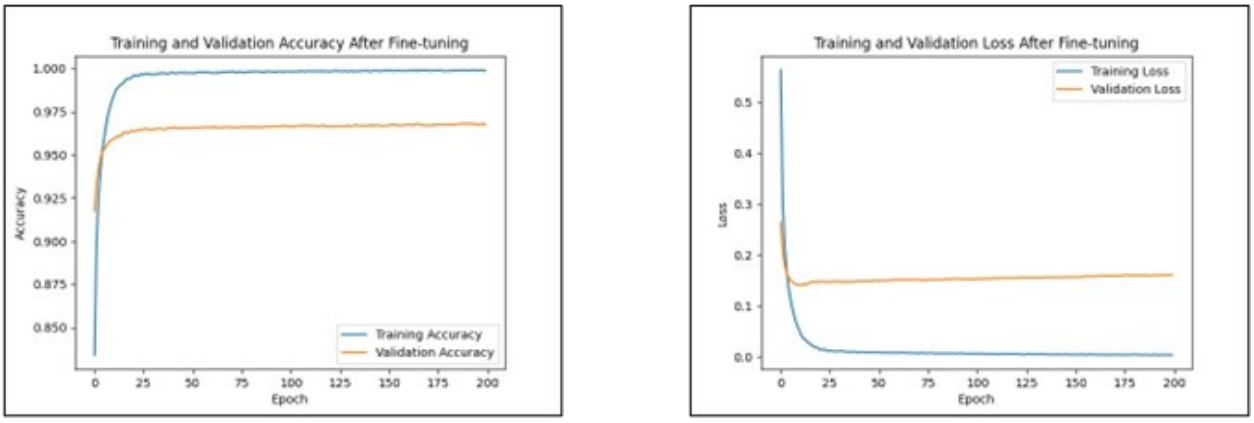
The proposed model was trained with the CIFAR-10 dataset along with the following training parameters:

- Epochs: 250 (50 Initial, 200 Fine tuning)
- Batch size: 32
- Activation Function: ReLU
- Optimizer: Adam
- Learning rate optimizer: ReduceLRONPlateau

The proposed model after training with CIFAR-10 dataset achieved an excellent validation accuracy of 96.7% and validation loss of 16.0% with an iteration time of 65 seconds per epoch. As a result, the model was also able to classify all the classes in CIFAR-10 accurately. Table 5.1 compares how the proposed model performed compared to baseline architecture on CIFAR-10. Figure 5.2 shows the performance of the proposed model on CIFAR-10 in terms of training accuracy, validation accuracy, training loss and validation loss. Figure 5.3 illustrates learning rate and Figure 5.4 shows two sample images from the dataset predicted and classified accurately by the model. Figure 5.5 represents the models confusion matrix with CIFAR-10.

Table 5.1. Performance metrics with Baseline and Proposed model on CIFAR-10

Architecture	Validation Accuracy (%)	Validation Loss (%)	Training Accuracy (%)	Training Loss (%)	Parameters (in millions)	Iteration time per Epoch (sec)
Baseline	94.3	37.1	99.8	0.8	4.0	126.0
Proposed	96.7	16.0	99.9	0.4	4.5	65.0



(a) (b)

Figure 5.2. Performance of Proposed model on CIFAR-10

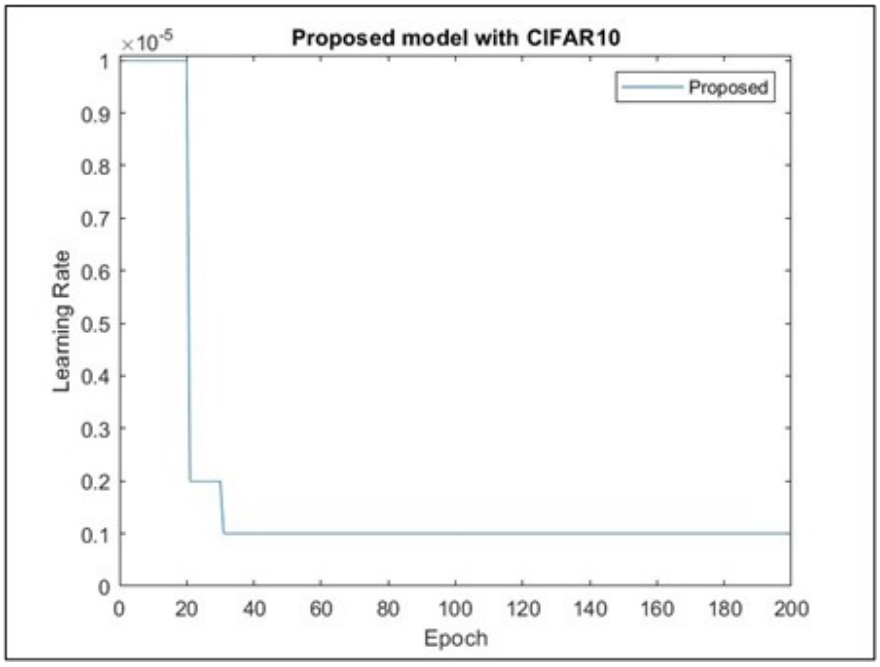


Figure 5.3. Learning rate with Proposed model on CIFAR-10

(a) (b)

Figure 5.4. Sample image class predictions by Proposed model on CIFAR-10

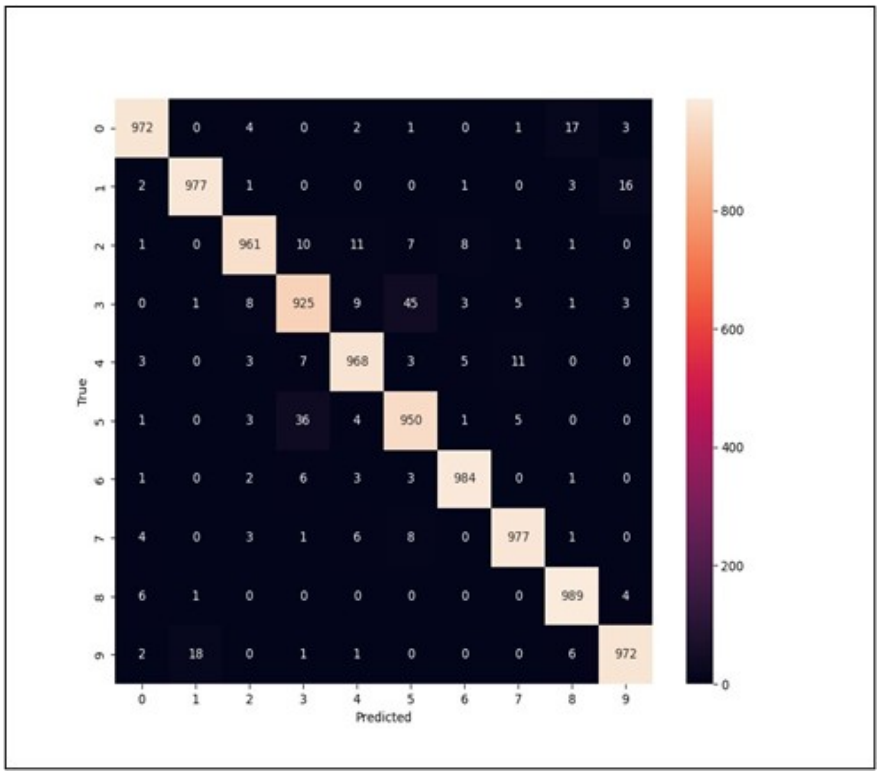


Figure 5.5. Confusion matrix with Proposed model and CIFAR-10

5.2.2 CIFAR-100 Dataset

CIFAR-100 dataset is similar to CIFAR-10 in that both datasets consist of 60000 (32x32) images with 50000 used for training the model while the remaining 10000 is used for testing. However, as the name suggests, CIFAR-100 consists of 100 classes and these classes along with their images are mutually exclusive to CIFAR-10 [28]. Figure 5.6 provides an overview of all classes in this dataset.

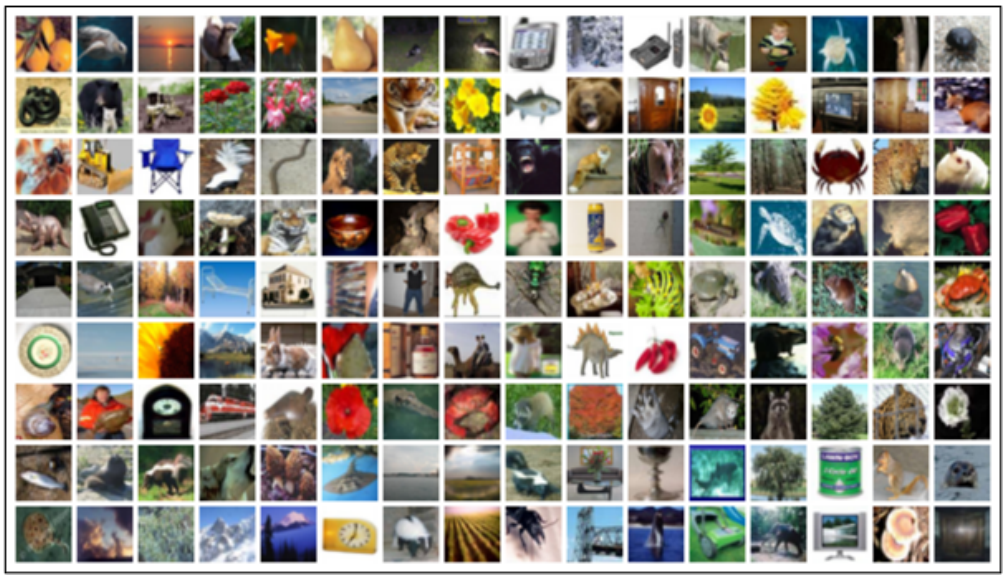


Figure 5.6. Image classes in CIFAR-100

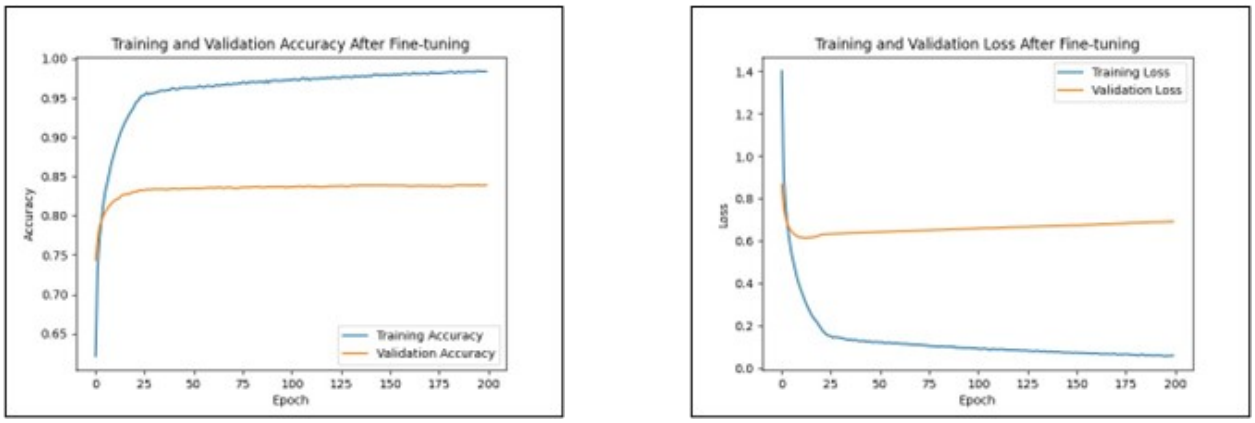
The proposed model was also trained with the CIFAR-100 dataset along with the following training parameters:

- Epochs: 250 (50 Initial, 200 Fine tuning)
- Batch size: 32
- Activation Function: ReLU
- Optimizer: Adam
- Learning rate optimizer: ReduceLRONPlateau

The proposed model after training with CIFAR-100 dataset achieved a validation accuracy of 83.9% and validation loss of 68.8% with an iteration time of 64 seconds per epoch. Similar to results with CIFAR-10, the model was also able to classify all the classes in CIFAR-100 accurately. Table 5.2 compares how the proposed model performed compared to baseline architecture on CIFAR-100. Figure 5.7 shows the performance of the proposed model on CIFAR-100 in terms of training accuracy, validation accuracy, training loss and validation loss. Figure 5.8 illustrates learning rate and Figure 5.9 shows two sample images from the dataset predicted and classified accurately by the model. Figure 5.10 represents the models confusion matrix with CIFAR-100.

Table 5.2. Performance metrics with Baseline and Proposed model on CIFAR-100

Architecture	Validation Accuracy (%)	Validation Loss (%)	Training Accuracy (%)	Training Loss (%)	Parameters (in millions)	Iteration time per Epoch (sec)
Baseline	76.9	172.7	99.2	2.4	4.0	126.0
Proposed	83.9	68.8	98.2	6.3	4.5	64



(a) (b)

Figure 5.7. Performance of Proposed model on CIFAR-100

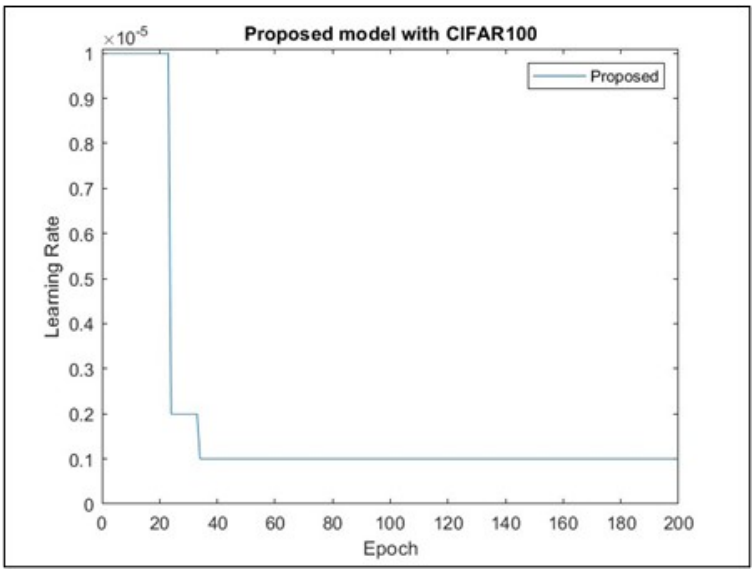


Figure 5.8. Learning rate with Proposed model on CIFAR-100

(a) (b)

Figure 5.9. Sample image class predictions by Proposed model on CIFAR-100

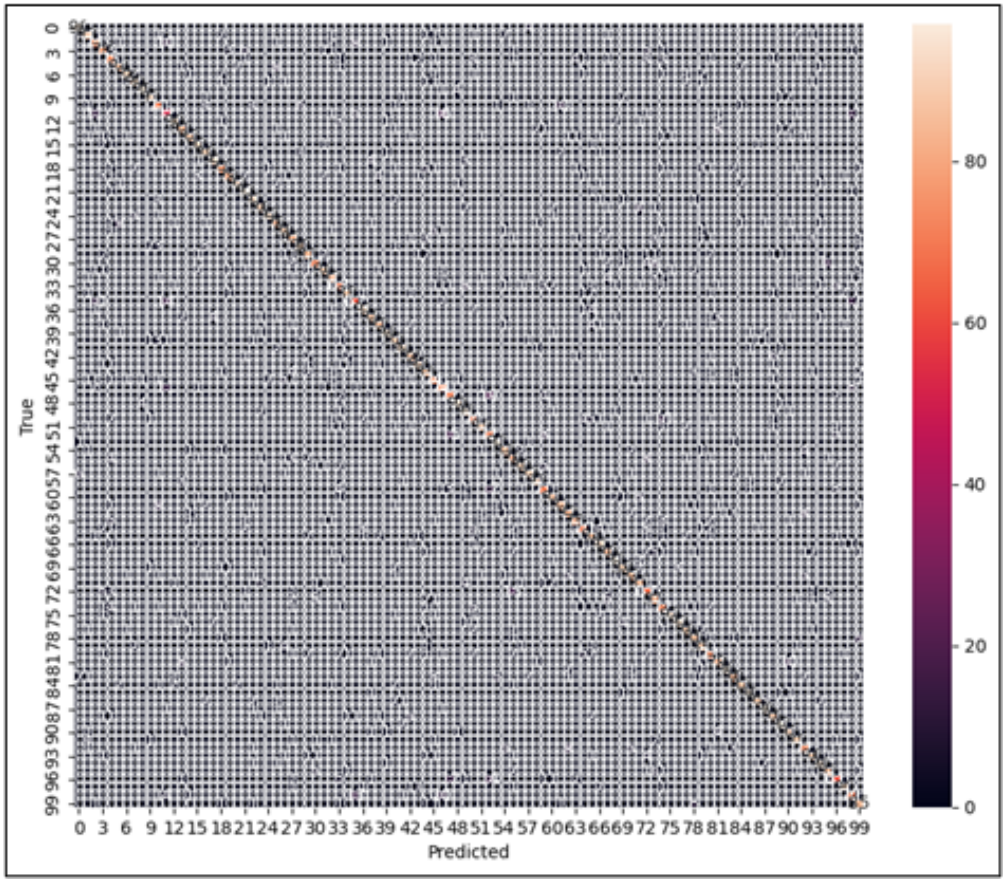


Figure 5.10. Confusion matrix with Proposed model and CIFAR-100

5.3 Summary

The proposed and baseline models were trained to classify images from CIFAR-10 and CIFAR-100 datasets. The results from the training are documented in Table5.3 and Figures 5.11 through 5.18. It is evident from the results that the proposed model outperforms the baseline model in all performance metrics with CIFAR-10 and all except training accuracy and loss with CIFAR-100. The improvement in accuracy, reduction in loss and iteration time stems from the addition of a new dense layer, implementation of dropout regularization, feature extraction and dual training phase as part of the training regime. The proposed model avoids overfitting as reflected in the superior validation loss reduction when compared to baseline. The model also is extremely efficient in that the iteration time is almost cut in

half when compared to baseline. Lastly, the model also exhibits robustness by being capable of performing well with both datasets.

Table 5.3. Performance metrics with Baseline and Proposed model on CIFAR-10 and CIFAR-100

Dataset	Architecture	Validation Accuracy (%)	Validation Loss (%)	Training Accuracy (%)	Training Loss (%)	Parameters (in millions)	Iteration time per Epoch (sec)
CIFAR-10	Baseline	94.3	37.1	99.8	0.8	4.0	126.0
	Proposed	96.7	16.0	99.9	0.4	4.5	65.0
CIFAR-100	Baseline	76.9	172.7	99.2	2.4	4.0	126.0
	Proposed	83.9	68.8	98.2	6.3	4.5	64

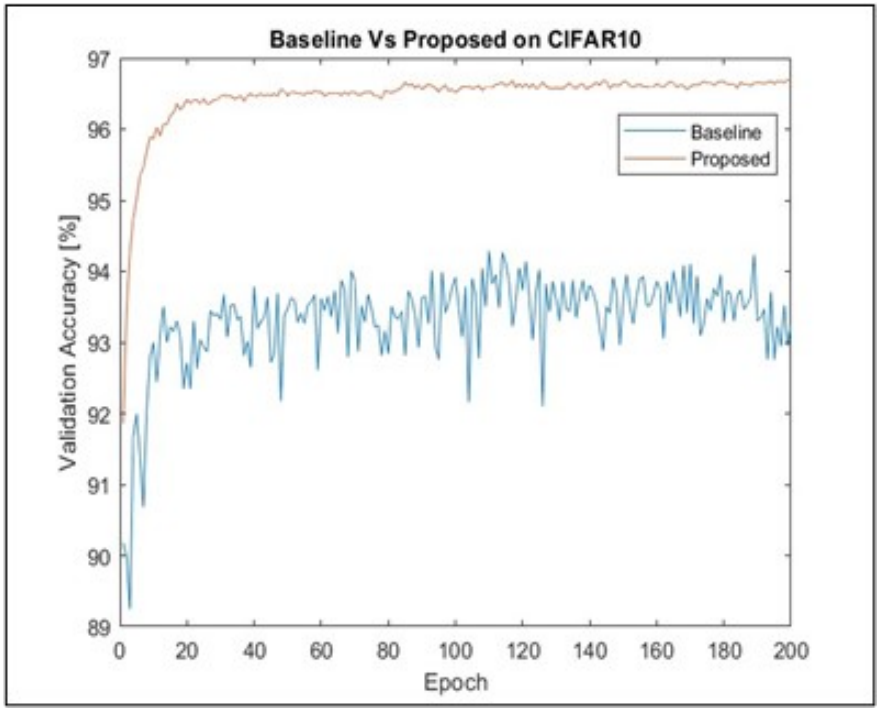


Figure 5.11. Validation accuracy of Proposed and Baseline models with CIFAR-10

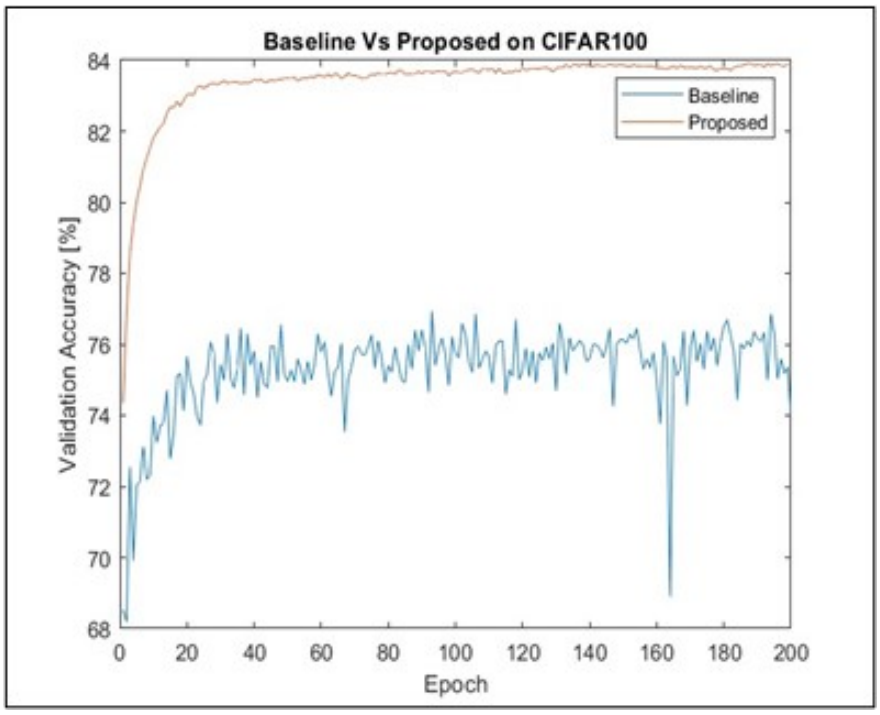


Figure 5.12. Validation accuracy of Proposed and Baseline models with CIFAR-100

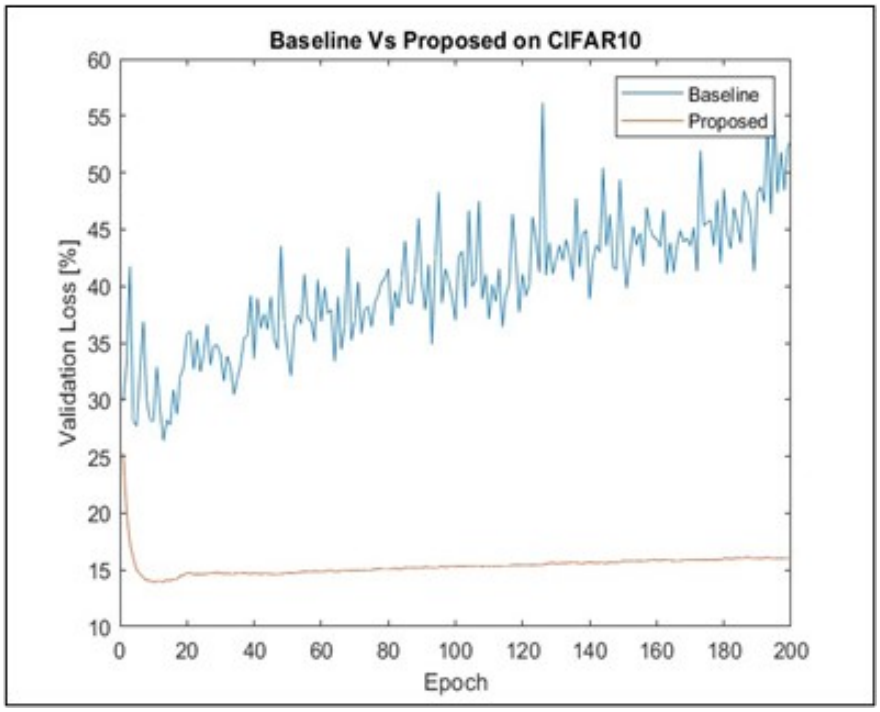


Figure 5.13. Validation loss of Proposed and Baseline models with CIFAR-10

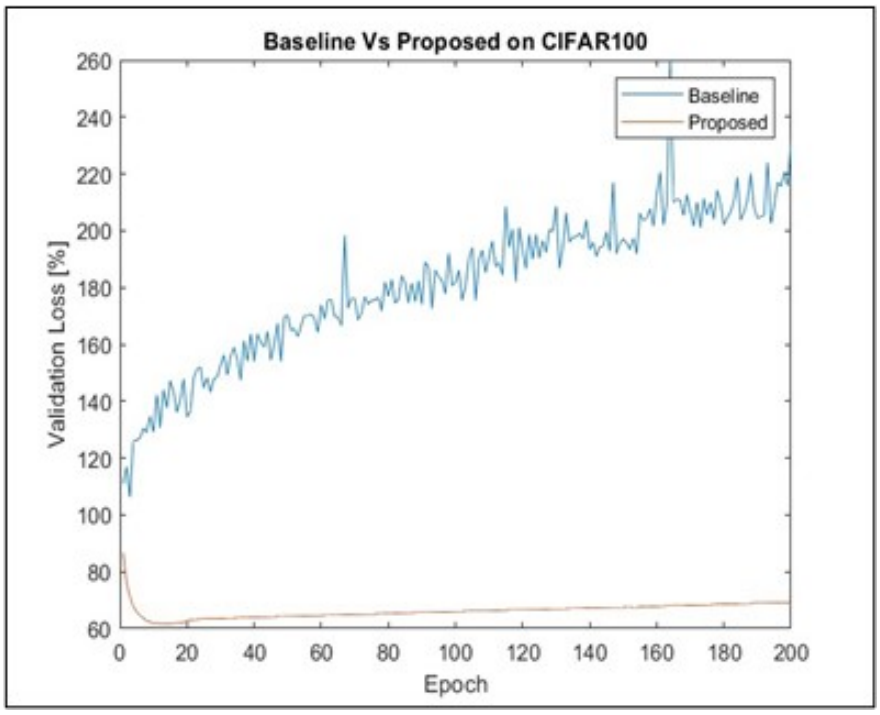


Figure 5.14. Validation loss of Proposed and Baseline models with CIFAR-100

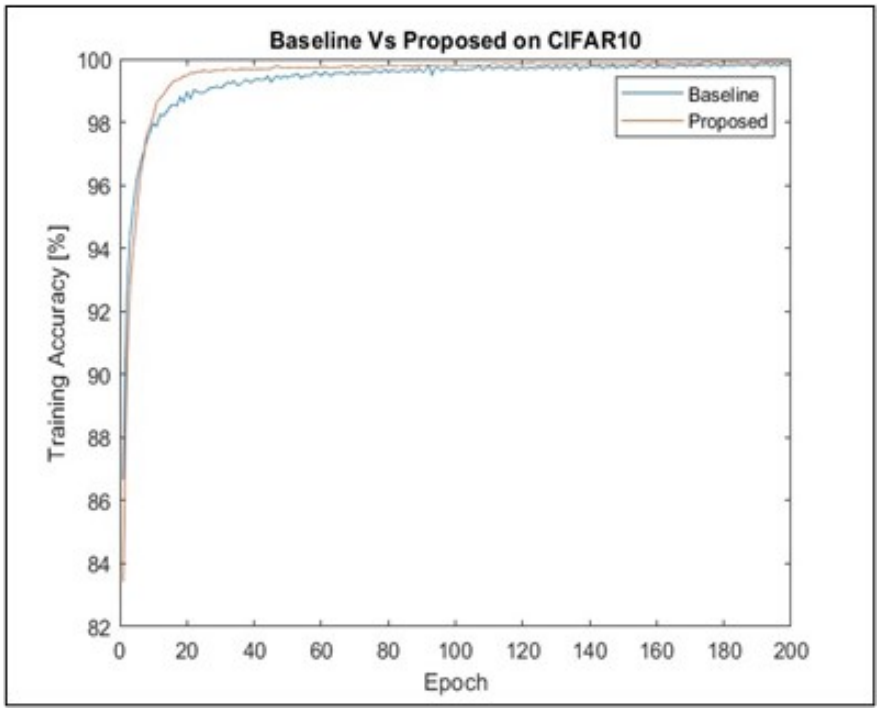


Figure 5.15. Training accuracy of Proposed and Baseline models with CIFAR-10

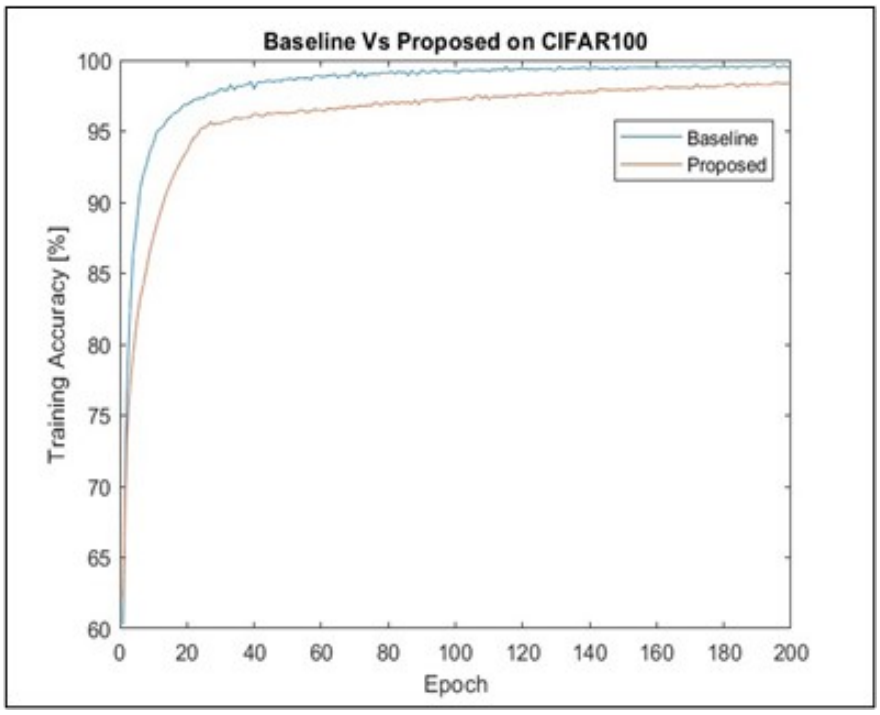


Figure 5.16. Training accuracy of Proposed and Baseline models with CIFAR-100

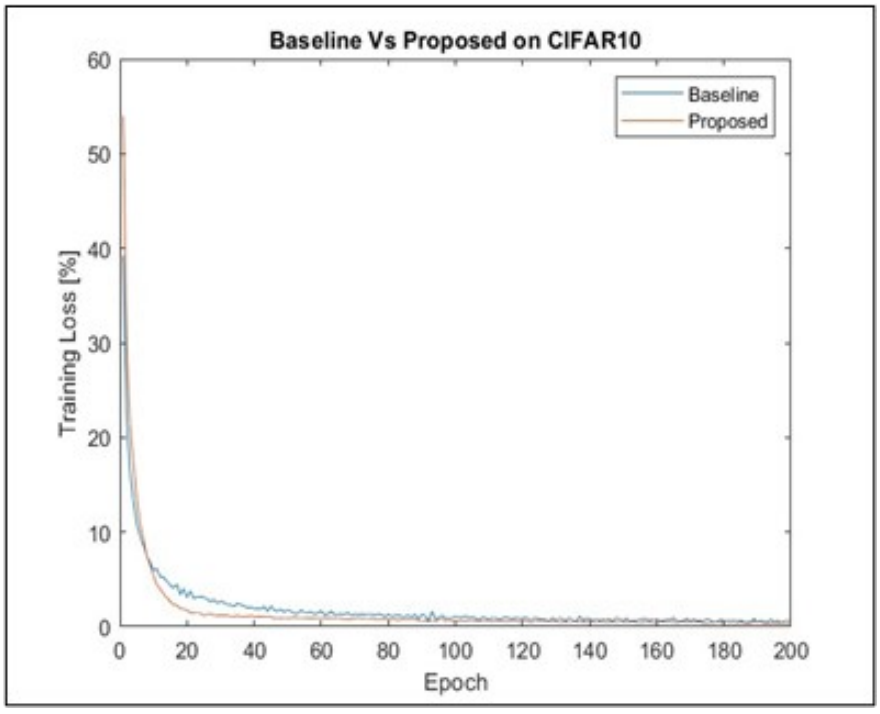


Figure 5.17. Training loss of Proposed and Baseline models with CIFAR-10

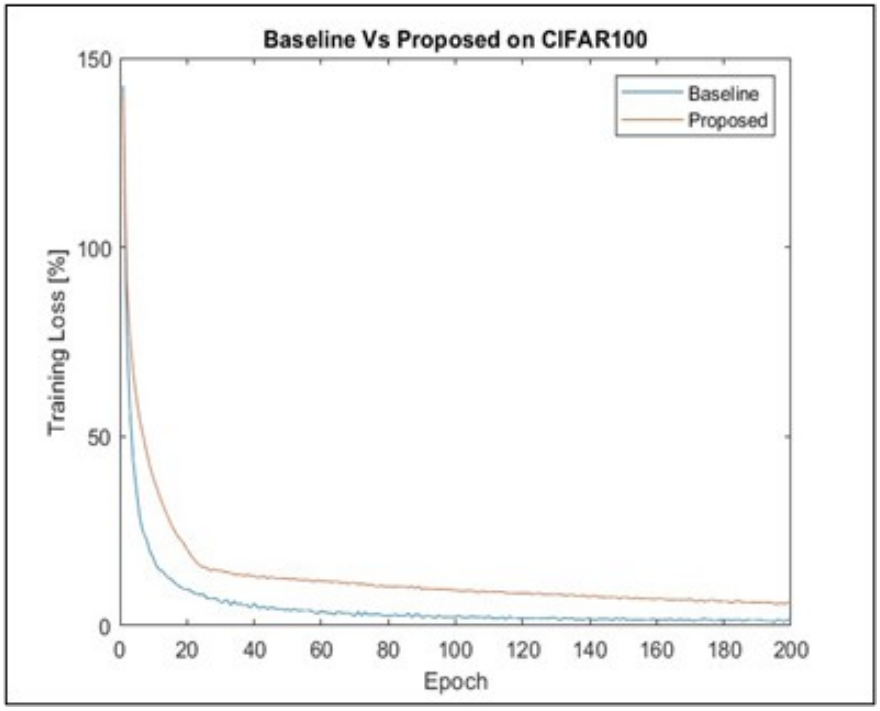


Figure 5.18. Training loss of Proposed and Baseline models with CIFAR-100

6. CONCLUSION

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, empowering a myriad of applications ranging from image classification to object detection and beyond. Traditionally, EfficientNetB0 has been a preferred choice among researchers due to its balanced efficiency and manageable model size, offering a well-rounded solution for a wide array of image processing tasks. Nonetheless, when subjected to rigorous testing across diverse benchmark datasets such as CIFAR-10 and CIFAR-100, this model has sometimes shown a tendency towards high validation loss, a symptom indicative of overfitting. This issue not only undermines the model's generalization to unseen data but also highlights the need for architectural innovation and refined training methodologies.

In response to these challenges, this thesis introduces an innovative architecture dubbed Enhanced Multiple Dense Layer EfficientNet specifically tailored for the task of image classification. This architectural evolution marks a departure from traditional designs by incorporating multiple dense and dropout layers, strategically engineered to enhance the model's accuracy and significantly curb validation loss. These additions are instrumental in bolstering the model's ability to generalize across different datasets, effectively mitigating the overfitting phenomenon and heralding a new era of CNN design optimized for robust performance.

A hallmark of the Enhanced Multiple Dense Layer EfficientNet is its remarkable efficiency in training. The model not only achieves high levels of accuracy but also demonstrates a substantial reduction in training time. This efficiency is attributed to the model's innovative structure and the employment of a carefully curated training regimen that accelerates convergence and diminishes validation loss. Such attributes make the model exceptionally suited for real-time applications, where swift, accurate decisions based on visual data are paramount.

Lastly, the Enhanced Multiple Dense Layer EfficientNet showcases versatile adaptability, suggesting its potential utility across a spectrum of image recognition challenges. The model's impressive performance on benchmark datasets like CIFAR-10 and CIFAR-100 serves

as a testament to its broad applicability and sets the stage for its exploration in various other domains.

7. FUTURE SCOPE OF WORK

As we journey further into the fascinating world of deep learning, the horizon for what comes next with the Enhanced Multiple Dense Layer EfficientNet is brimming with exciting possibilities. This model, having shown impressive results on CIFAR-10 and CIFAR-100, stands at the cusp of diving into even more complex and larger datasets. The path ahead includes tinkering with the model's hyperparameters and structure, akin to fine-tuning a high-performance engine, ensuring every part works in perfect harmony for peak efficiency, accuracy, and adaptability. Prospective researchers can harness advanced methods like Neural Architecture Search (NAS) to meticulously tailor this model's architecture to make it even more efficient and accurate. Additionally, researchers can enhance the model with the latest in CNN components attention mechanisms that allow it to focus and novel activation functions that could unlock new learning potentials. These enhancements provide the model with a new set of sophisticated tools to better understand and interpret the visual data it encounters. In the quest for a model that not only learns but generalizes well, researchers can explore beyond the familiar terrain of dropout techniques, venturing into the realm of L1/L2 regularization and innovative methods like Mixup or CutMix. These strategies are allies in the fight against overfitting, ensuring the model can perform admirably across diverse and complex datasets. Lastly, techniques like model quantization, pruning, and knowledge distillation can reduce the model size without significant impact to efficiency and accuracy. Once successfully implemented, the model can take a huge step forward in making advanced technologies accessible right at the edge in smartphones, smartwatches and embedded systems where computational resources are at a premium.

REFERENCES

[1] M. Alfahdawi, K. M. Ali Alheeti, and S. Al-Rawi, "Intelligent object recognition system for autonomous and semi-autonomous vehicles," Jun. 2021, pp. 227–233. DOI: 10.1109/ICICT52195.2021.9568417.

[2] K. Srinivasan, A. Raman, and R. Kvs, "Leveraging computer vision for emergency vehicle detection-implementation and analysis," Oct. 2020. [Online]. Available: https://www.researchgate.net/publication/344881843_Leveraging_Computer_Vision_for_Emergency_Vehicle_Detection-Implementation_and_Analysis.

[3] L. Zhou, Z. Qin, P. Xia, *et al.*, "General model for manufacturing defect detection crossing multiple products," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, 2023, pp. 1595–1600. DOI: 10.1109/ICMLA58977.2023.00241.

[4] S. Shetye, S. Shetty, S. Shinde, C. Madhu, and A. Mathur, "Computer vision for industrial safety and productivity," in *2023 International Conference on Communication System, Computing and IT Applications (CSCITA)*, 2023, pp. 117–120. DOI: 10.1109/CSCITA55725.2023.10104764.

[5] L. R. Kennedy-Metz, P. Mascagni, A. Torralba, *et al.*, "Computer vision in the operating room: Opportunities and caveats," *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 1, pp. 2–10, 2021. DOI: 10.1109/TMRB.2020.3040002.

[6] S. Ding, H. Zhao, Z. Han, and Y. Tang, "A method of cell counting based on computer vision," in *2022 12th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2022, pp. 729–734. DOI: 10.1109/CYBER55403.2022.9907162.

[7] C. Rakshitha and C. Selvan, "Improving cctv footage in computer vision using deep learning techniques," in *2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS)*, 2023, pp. 1–5. DOI: 10.1109/ICIICS59993.2023.10421193.

[8] S. Saha, S. Banerjee, S. Dhar, A. Bandyopadhyay, S. Saha, and S. Ray, "Unmanned aerial vehicle for district surveillance with computer vision and machine learning," in *2023 7th International Conference on Electronics, Materials Engineering Nano-Technology (IEMENTech)*, 2023, pp. 1–6. DOI: 10.1109/IEMENTech60402.2023.10423530.

[9] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y.

[10] H. A. Kassir, N. V. Kantartzis, P. I. Lazaridis, *et al.*, "Improving doa estimation via an optimal deep residual neural network classifier on uniform linear arrays," *IEEE Open Journal of Antennas and Propagation*, vol. 5, no. 2, pp. 460–473, 2024. DOI: 10.1109/OJAP.2024.3362061.

[11] Z. Wang, C. Tang, X. Sima, and L. Zhang, "Research on application of deep learning algorithm in image classification," in *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2021, pp. 1122–1125. DOI: 10.1109/IPEC51340.2021.9421185.

[12] P. Ramakrishnan, K. Dhanavel, K. Deepak, and R. Dhinakaran, "Autonomous vehicle image classification using deep learning," in *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, 2023, pp. 97–104. DOI: 10.1109/ICSCDS56580.2023.10104830.

[13] Charleen, C. Angelica, H. Purnama, and F. Purnomo, "Impact of computer vision with deep learning approach in medical imaging diagnosis," in *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)*, vol. 1, 2021, pp. 37–41. DOI: 10.1109/ICCSAI53272.2021.9609708.

[14] G. G, J. Thimmiaraja, C. J. Shelke, G. Pavithra, V. K. Sharma, and D. Verma, "Deep learning with unsupervised and supervised approaches in medical image analysis," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2022, pp. 1580–1584. DOI: 10.1109/ICACITE53722.2022.9823491.

[15] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, 2007. DOI: 10.1080/01431160600746456. [Online]. Available: https://doi.org/10.1080/01431160600746456.

[16] Q. Wang, R. Zhou, and Y. Yan, "A two-stage approach to note-level transcription of a specific piano," *Applied Sciences*, vol. 7, no. 9, p. 901, 2017. DOI: 10.3390/app7090901. [Online]. Available: https://doi.org/10.3390/app7090901.

[17] P. Dileep, D. Das, and P. K. Bora, "Dense layer dropout based cnn architecture for automatic modulation classification," in *2020 National Conference on Communications (NCC)*, 2020, pp. 1–5. DOI: 10.1109/NCC48643.2020.9055989.

[18] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 0588–0592. DOI: 10.1109/ICCSP.2017.8286426.

[19] S. Saha, "A comprehensive guide to convolutional neural networks the eli5 way," *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[20] P. Kalgaonkar and M. El-Sharkawy, "Nextdet: Efficient sparse-to-dense object detection with attentive feature aggregation," *Future Internet*, vol. 14, no. 12, 2022, ISSN: 1999-5903. DOI: 10.3390/fi14120355. [Online]. Available: https://www.mdpi.com/1999-5903/14/12/355.

[21] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html.

[22] M. Tan and Q. V. Le, *Architecture of efficientnet b0 with mbconv as basic building blocks*, ResearchGate, 2019. [Online]. Available: https://www.researchgate.net/figure/Architecture-of-EfficientNet-B0-with-MBConv-as-Basic-building-blocks_fig4_344410350.

[23] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 10 096–10 106. [Online]. Available: https://proceedings.mlr.press/v139/tan21a.html.

[24] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *J Big Data*, vol. 8, no. 1, p. 53, 2021. DOI: 10.1186/s40537-021-00444-8. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8.

[25] M. Pendse, V. Thangarasa, V. Chiley, R. Holmdahl, J. Hestness, and D. DeCos, "Memory efficient 3d u-net with reversible mobile inverted bottlenecks for brain tumor segmentation," *Journal of Medical Imaging*, vol. 10, no. 1, p. 015 001, 2023. DOI: 10.1117/1.JMI.10.1.015001. [Online]. Available: https://doi.org/10.1117/1.JMI.10.1.015001.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.

[27] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[28] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical Report, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.