Object Recognition using Machine Learning

Bhut Vandit Shamjibhai
2338357

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Master of Science

Master of Science

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

September 30, 2024

MSc dissertation

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed Vandit S. Bhut (candidate)

Date 30/09/2024

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed Vandit S. Bhut (candidate)

Date 30/09/2024

Statement 2

I hereby give my consent for my thesis, if accepted, to be made available for photocopying and inter-library loan, and for the title and summary to be made available to outside organisations.

Signed Vandit S. Bhut (candidate)

Date 30/09/2024

Abstract

Its ever-growing importance in current computer vision and machine learning (ML) research cannot be ignored, especially in issues of practical relevance such as autonomous driving, detection of diseases, and decision-making. This project compares the performance of three convolutional neural (CNN) architectures (EfficientNet, DenseNet, and a traditional CNN) to classify images on the CIFAR-100 dataset, an image benchmark that can be challenging for computer classification algorithms. After evaluating three algorithms, this study also demonstrates the effectiveness of using majority voting, as well as the same weights and different weight majority voting.

The core focus of this study is to assess the Efficiency, complexity, and classification performance of this architecture with a view to determining the possible compromises that exist between the complexity of the model and the performance trade-offs. The compound scaling architecture known for the EfficientNet family seeks to achieve the goals of high accuracy while taking optimal computational resources. DenseNet enhances the reuse of features and the flow of gradients but has the disadvantage of dense connections that tend to increase the computation costs. Conventional CNN is a simple model that can be employed as the type to be compared with the more sophisticated networks.

This study examines how these models deal with the complex requirements of real-world image datasets. It looks at factors such as overfitting, generalisation to new data, and inference time through extensive experimentation and analysis. The results highlight the strengths and weaknesses of each architecture, offering valuable insights into their efficacy and practical utility in object recognition tasks.

In contrast, it is worth noting that such new generation architectures, as Enabled EfficientNet or DenseNet provided better results, but have some limitations that need to be weighed up, particularly where there are limited computing resources.

Acknowledgements

I am grateful to Professor Xiaochun Cheng, my supervisor, for making this work possible. His guidance and advice carried me through all the stages of writing my project.

I am also thankful to the Swansea University Department of Computer Science for the facilities and software used to complete this work.

I would also like to give special thanks to my family for their continuous support and understanding when undertaking my research and writing my project.

Finally, I would like to thank God, for letting me through all the difficulties. I have experienced your guidance day by day. You are the one who let me finish my degree I will keep on trusting you for my future.

Table of Contents

Chapter 1

Introduction

In the current digital age, we are crowded with data so, there is a growing need for advanced, fast and Artificial Intelligence (AI) intelligence analysis techniques. Those things enable significant advancements in image classification, object detection, and image generation [18]. A considerable proportion of research in the field of computer vision and machine learning is performed on the CIFAR-100 dataset [18]. This investigation is aimed at proving this hypothesis. Alternatively, preparing an accurate image is one of the most challenging jobs in computer vision. This implies the existence of certain categories out of which an image must be placed into one of them [18].

The most common architecture for image classification tasks is Convolutional Neural Networks. CNN has ability to automatically learn hierarchical features from raw pixel data. In this research, we compare two CNN architectures, namely DenseNet and EfficientNet, with a conventional CNN model [18]. EfficientNet is one model that can adjust in size. It maintains balance among depth, width, and resolution to enhance precision and computational effectiveness. DenseNet establishes robust inter-layer connections. This facilitates feature reuse and enhances the model's learning process.

Majority voting is an ensemble technique often used to improve model performance by combining predictions from multiple classifiers [18]. In this approach, each model makes a prediction, and the final output is determined by the class label that receives the most votes [18]. This method helps to reduce individual model biases and variance, leading to more robust and accurate predictions.

This study aims to compare the performance of these models on the CIFAR-100 dataset, evaluating their classification accuracy, computational complexity, and robustness in handling a challenging dataset with many categories [18]. we aim to provide insights into how modern CNN architectures perform in a competitive image classification task and to find the trade-offs between accuracy and computational efficiency by using the strengths CNN model likely as EfficientNet, DenseNet and classical CNN models [18].

The report commences with the introduction which gives the outline of the purpose of the research, the background of the study, the relevant literature related to the research, and one database – the CIFAR-100 [18]. It then proceeds to detail the structures of the assembled networks of the CNN, EfficientNet and DenseNet. The report presents the approach for training and analysing the models [18]. Afterwards, it analyses and

compares the results across the different models, focusing the primary finding and its consequences for next classification projects.

1.1 Motivations

The motivation behind this research is to renew and improve object recognition techniques and methods in relation to challenges arising in the 21st century. The conventional object recognition methods are no longer considered adequate because of the huge volume, velocity, and variety of data sources being experienced [24]. The domain of ML and AI offers a wonderful opportunity to realise improvement within object recognition technologies [22].

This study aims to discover and reveal the hidden patterns or, rather trends within large datasets that might not be easily visible using other conventional methods of analysis [29]. In addition, this research looks to enhance the analytical power of object recognition by integrating machine learning algorithms critical in the identification of patterns and trends in data. Therefore, using this model, one can enhance the analytical skills [22].

The main objective of this research is to overcome the inefficiencies of traditional techniques when dealing with large and heterogeneous[1] datasets. Traditional methods are not capable of handling big datasets that comprise multi-type data. In this regard, we enhance these traditional techniques by incorporating ML algorithms to develop better efficiency in handling complex datasets [29]. We aim to use ML algorithms to reduce the complexity faced while processing huge and heterogeneous datasets.

Additionally, this model helps predict possible dangers and challenges that might impact a business or an entire nation, enabling proactive measures to reduce such risks and protect employment for employees and the national economy. Essentially, this approach helps different stakeholders forecast the future [22]. The government will anticipate weather patterns, assess the country's economic development, and much more. Additionally, private enterprises will forecast financial metrics concerning their growth and various other aspects [29].

This research can be attributed to the need to update and improve intelligence analysis techniques to cope with the current challenges. In other words, the primary focus is to modernise and empower analysis methods to meet the demands of present-day scenarios [22]. Companies and people in positions of power make decisions for the public and those economically disadvantaged [29].

[1] Heterogeneous most generally means consisting of different, distinguishable parts or elements

1.2 Objective

The objective of this research work is basically to develop a strong, flexible machine learning model for object recognition, while the essential component in identifying algorithms should be suitable for practical applications [22]. The study focuses on the comparison between conventional CNN, EfficientNet, and DenseNet with regards to their effectiveness in the solution of CIFAR-100 image classification [18]. Several key factors will be considered in the evaluation, such as class accuracy and the capability of the models to remember complex and diverse image datasets.

Additionally, this research will explore the integration of majority voting as an ensemble technique to enhance model performance. The majority voting mechanism is expected to yield more robust and accurate results by combining the predictions of the conventional CNN, EfficientNet, and DenseNet architectures [22]. This approach will help mitigate individual model weaknesses and improve overall classification precision, making the model more adaptable for real-world object recognition tasks.

Furthermore, this work tries to complement the various limitations identified with the use of traditional analytical approaches. It is envisaged that the model derived will be useful in decision-making for ideal real-world activities and challenges. This work intends to further advance this understanding of the behaviour of contemporary CNN designs to challenging image datasets and shall help in selecting appropriate models for streamlined and efficient image classification tasks [29].

1.3 Overview

The rest of this report is organised as follows: Chapter 2 comprehensively reviews and critically discusses the related literature on core machine learning, object recognition concepts, and the evolution of CNNs, with special attention to the EfficientNet and DenseNet architectures.

Chapter 3 presents the software used during the experiment. That means it includes research on programming languages, libraries, and frameworks used to develop and execute machine learning models. Optimisation and regularisation methods are fully explained to show how the performance of the proposed models can be enhanced.

The Chapter 4 will explain the CIFAR-100 data set in terms of the configuration, the hindrances it involves, and its use for image classification. This chapter will set a broader context in which the issue that is problematised and pursued by the machine learning models exists.

The Fifth Chapter presents the methodological consideration that was taken in the conduct of the research exploration. This addresses every step within the processes of

design on the model architecture of the CNN, DenseNet and EfficientNet models, data preparations and training.

The majority theory, which is decision rule based and developed on the idea of False Positive and False Negative aspects, is discussed extensively in the report chapter six. It reveals all factors that serve to influence decision taking such as accuracy, False Positive Rate, and False Negative Rate.

The seven chapter compares the effectiveness of each model with consideration to the factors of accuracy and the accuracy and performance of the implementation processes. From such analysis of the advantages and disadvantages, a deeper comprehension is achieved on the respective opportunities that present themselves in the use of varied CNN architectures to specific image dataset problems.

Therefore, Chapter 8 provides a summation of the report's findings and proposes some avenues that can be pursued in future studies. Thus, this section brings in a summary of the major inputs into the recent study and prepares the ground for the next step.

Chapter 2

Background and Literature Review

The section below entails fundamental knowledge on the subject, including algorithms and other technologies that relate to it. It is also organised into basic concepts, machine learning, object detection, CNNs, EfficientNet, DenseNet, applications, and several comparative analyses [39].

The main task in the domain of computer vision is image classification, which has reached a high stage of development [57]. It is indispensable in many practical tasks, such as face recognition, self-driving cars, diagnosis, and recommendation systems [39]. At the heart of recent success in image classification has been the development of deep learning architectures, where especially Convolutional Neural Networks have reached the highest performance by acquiring the capability of autonomous feature extraction at higher levels from the raw images [39].

The CIFAR-100 represents a well-known benchmark for evaluating image classification models. Compared with the earlier released version of this dataset, CIFAR-10, which has only ten different classes, it is substantially more complex [57]. CIFAR-100 embodies a difficult challenge due to strong class variability, reduced dimensionality, and subtle differences that need to be addressed for accurate classification of images [57].

2.1 Machine learning

Machine learning is a subset of artificial intelligence which enables a machine or computer to operate by using knowledge based on past data and not through a sequence of predetermined instructions [63]. To develop a certain machine-learning algorithm, one must first upload the set of sample data which the algorithm is supposed to replicate (otherwise known as training data), and afterward, it will proceed to generate foresight or locate relationships in patterns on fresh information. Therefore, the algorithm is replicated by showing whatever it is that one wants it to learn [63].

Some methods or algorithms of learning known as tree classification, neural networks models and support vector machines are used in the sensing domain by understanding various patterns in data [63].

A machine learning model learns from training data, which is a dataset of input data paired with correspondingly correct answers [63]. Machine learning models require training data to learn; after training on the data, the learning machine algorithm creates

a model capable of inferring or assessing the probability of new data [41]. If the model has been built with the needed data without any further costs being involved, it becomes simple to produce and act loyally to make predictions or draw conclusions [41]. We can distinguish the acquisition of knowledge into three main categories: supervised, unsupervised and reinforcement learning [41].

To sum up, the goal of machine learning is to create algorithms that allow computers to learn and behave in ways not directly programmed into them, instead of finding and specifying all the things they possibly do [41].

2.2 Object recognition

Object recognition, itself a subcategory of computer vision, refers to the methods used for finding and classifying objects within an image or video [35]. This project work focused solely on images. Object recognition allows the machine to find certain objects or object categories in visual data by finding such things as cars, persons, toys, or animals in a photograph [35]. It is a critical ingredient of many AI-powered applications, including but not limited to self-driving cars, augmented reality, and robotics [35].

Object detection and object recognition represent major tasks of the area of computer vision [48]. While object detection does not include only finding objects inside an image but finding the exact one using bounding boxes and labels, object recognition deals with identification of what is the object without finding it inside the image [35]. Feature extraction for major patterns within an image and image classification is done through machine learning models [35]. These have included object recognition and detection methods such as YOLO and Faster R-CNN in a combination to realise real-time identification and localisation of multiple objects [48].

However, the paper focused on CNNs and their two different architectures. The uses of these technologies run deep, starting from autonomous vehicles and security systems to medical image processing and retail automation [48]. Object recognition is key to intelligent perception and interaction with the environment through machines [48].

The traditional CNN architectures have gained enormous success for image classification benchmarks, but to extend this to the more challenging datasets of CIFAR-100, high performance and efficient models needed to be designed [35]. Such high-performance models include EfficientNet and DenseNet-two of the most popular model architectures due to their novel design paradigm, aiming at mitigating most of the familiar challenges of deep learning: model scalability, efficiency of parameters, and the vanishing gradient problem [48].

2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specific type of deep neural networks employed to carry out deep learning on data with a grid-like topology, for example, images and other multi-dimensional matrices [8]. Convolutional layers form the foundation of a CNN and help in extracting properties like edges or textures from images. This technology is rooted in the work by LeCun et al. (1998) and their so-called development of the LeNet-5 architecture [8].

The process applies a small-sized filter, usually referred to as the kernel, across an input image to come up with feature maps representative of major visual properties [49]. Further, the feature maps are subjected to processing through successive convolutional and pooling, and full-connected layers to get the final classification output [49].

Convolutional Neural Networks have been used successfully on a wide range of image classification problems, especially the simpler ones, such as MNIST, CIFAR-10, CIFAR-100, and ImageNet [46]. These architectures, including AlexNet by Krichevskys et al. (2012), VGGNet by Simonyan & Zisserman (2014), EfficientNet, DenseNet, and ResNet by He et al. (2016), form important steps in the development of CNNs since it became possible to build deeper and more complex networks with better performance [46].

However, as network depth increases, a common issue of CNNs involves overfitting and the vanishing gradient problem, which may reduce the capability of a network to converge during training [49]. This has inspired more advanced architectures that address these limitations with the least sacrifice in terms of accuracy and computational efficiency [8].

2.4 EfficientNet

EfficientNet Introduced in 2019 by Tan and Le, it stands for a new scaling method for Convolutional Neural Networks [19]. There was only one way to make a more powerful model, and that meant going deeper (more layers), wider (more channels), or images with higher resolution [19]. This is often counterproductive since increased model size often produced rapidly diminishing returns in terms of improvements in accuracy [19].

This is what EfficientNet changed by coming up with something it called compound scaling. Instead of scaling one aspect of the network, it scales three: depth, width, and resolution, together in a balanced and efficient manner. The method allows the model to have more power without computational waste [19].

Members of the EfficientNet family, from B0 to B7, have been breaking new records and setting ultramodern results on image classification tasks, with improved benchmark records on the standard benchmark datasets such as ImageNet [19]. Surprisingly, these models do so with a lot fewer parameters and computations (FLOPs) compared to their precursors. Therefore, EfficientNet is particularly suited for resource-constrained applications like those in mobile devices or embedded systems [33].

As different studies have shown, this balanced scaling approach adopted by EfficientNet improves both the model accuracy and its generalisation capability across different datasets [33]. EfficientNet gives much more reliable and efficient deep learning with only a few resources, avoiding common issues of overfitting and underfitting often occurring when scaling just one part of the network [33].

2.5 DenseNet

DenseNet by Huang et al. (2017) was the first to leave from the typical CNN structure by proposing connecting many layers densely in a manner that allows passage of input and output activations between them [66]. Normally in CNN, each layer only receives input from the previously one layer so that it works in a bottom-up fashion. However, DenseNet links all layers so that the input from each layer has an output that can be fed to another layer [66].

The major advantages of DenseNet are induced by its dense connectivity pattern. First, it enhances the Gradient Flow problem of vanishing gradient lessens, and gradients can flow without much obstruction through the network, even in very deep architectures [1]. Second, it promotes Feature Reuse in that every layer has access to the feature maps of all earlier layers, hence yielding more compact and efficient models. Lastly, DenseNet is parameter-efficient compared to traditional deep CNNs; there is no need to learn redundant information for each layer despite its dense connectivity [66].

DenseNet outperformed the state of the art on several image classification benchmarks, including CIFAR-10 and CIFAR-100, especially in settings where one required a deeper network, but computational resources were significantly restricted [1]. Its novelty in feature reusing and dense connectivity influenced the design consideration in subsequent deep-learning models [1].

2.6 Summary

In the literature review, values such as DenseNet and EfficientNet are strides towards the design of a modern CNN that conquers prevailing issues like overfitting, vanishing gradients, and poor scaling. It is apparent from their results on exceedingly difficult datasets like CIFAR-100 that these solutions are of practical use where the balance of accuracy and computational cost has been teetering [1]. This work builds on these observations by focusing on the direct comparison of these architectures on the CIFAR 100 dataset and linking them to the modern image classification problem, the anatomical composition of each of them.

Chapter 3

Tools and Technologies

This section includes those tools and technologies that were used in the project. Such information is divided into: Programming Languages, Libraries and Frameworks, Optimisation and Regularisation, Data Augmentation and System Utilities. Here are some heuristics of the tools and technologies used in this project.

3.1 Programming Languages

3.1.1 Python

Python has been selected to be the main IT (Programming) language of the current project owing to the fact that it is easy to learn, flexible and it has rich libraries for data science, machine learning and deep learning [59].

As a general-purpose programming language, most scientists from different disciplines appreciate Python for its ease of use and efficiency in deploring machine learning and data interrogation skills [59]. It is simple enough to create programs that achieve advanced functionality. The prominent level of abstraction allows concentrating on the problem decoration rather than the machine language [59].

Machine learning programming languages such as python offer several libraries and framework for machine learning implementation which include NumPy, Matplotlib, SciPy, Pandas, Scikit-learn, TensorFlow, and Keras [59]. It gradually became the most efficient tool within the project due to the ease of use, practicality, and extensive use of libraries for data and machine learning [59].

3.2 Libraries and Frameworks

Machine Learning offers an incredible statistical and analytical industry backed up by efficient implementation of the strategies due to the use of different libraries and frameworks [38]. For instance, NumPy gives a range of tools for high-performance numerical computations, Pandas makes operations with data and its computation easy, whereas Matplotlib and Seaborn create beautiful pictures of data [38].

Whereas for more complicated tasks of machine learning and deep learning available libraries such as Scikit-learn ease the implementation of traditional machine learning models while TensorFlow and Keras are tailored towards the construction and training of deep neural networks [38]. Furthermore, SciPy has benefits in carrying out intricate scientific and arithmetic tasks achieving a complete picture of what Python offers in

technical areas. More information about the libraries is outlined in the next paragraphs [38].

3.2.1 TensorFlow

TensorFlow is an open-source machine learning framework created by Google which is also quite rich in features and is easy to adapt to other domains uplifting and educating people all over the world [55]. It has matured into one of the most versatile platforms for building as well as training deep learning models across different applications [55].

The reputation of TensorFlow grows due to the fact, that it can combine low level functionality, such as Tensors operations, with high level Templated classes for creation of complex nested structures as CNNs, RNNs or transformer-based models [55]. This is due to the excellent features of the framework in handling large models that require distributed computing and high-end hardware acceleration GPU and TPU which help practical deep learning training at scale, that is, accommodating big datasets [55].

Furthermore, TensorFlow includes active support of distributing models after their training to different environments such as mobile and cloud platforms and has the possibility of embedding such tools as the Tensor board which helps in showing the dynamics of training and makes it easy to check for bugs [55]. Such features as the reasonably prominent level of deep learning, scalability, and performance optimisation – make TensorFlow a reference project for efforts where creative machine learning models is needed, for instance, CNNs for complex image classification tasks, which are also the goals of this project [55].

3.2.2 Keras

Keras is a high secondary neural network API which simplifies deep learning model development in such a way that it can be used with less time than using the code library Keras is an open source neural network application PLC which operates on other frameworks and it allows the user to build complex networks in seconds, due to the fact that Keras wraps these low level operations and provides them through a user friendly interface [55]. This multipurpose deep learning API is efficient with complicated tasks like image classification, speech recognition, sentiment analysis and prediction of events over time or over a specific duration [55].

Keras is however quite flexible allowing users to edit the plans optimisers loss components and other aspects of the application while reaping the advantage of TensorFlow’s muscle [55]. This middle content has made it possible for the organisations to save time and focus on the models as some of the common activities associated with model development such as data preparation augmentation and evaluation are already built into the frame [55].

The first advantage of using Keras in this project is the fast-tracking process of developing deep learning system designs making it possible for designers to quickly construct and train neural networks like the variants of CNN that are studied in this document [55]. It is also worth noting that both community and documentation assistance are extensive and hence reliable put it simply [55].

3.2.3 Scikit-learn

Scikit-learn, despite being one of the many machine learning libraries found in Python, stands out most in its simplicity, dependability, and provision of most of the conventional machine learning tools [43]. In this area, many effective approaches like imputation of missing data, normalisation of feature values, and encoding of categorical variables are provided [43]. Also in this library are numerous algorithms for supervised as well as unsupervised learning such as regression and classification, clustering, dimensionality reduction and many more [43].

In this project, Scikit-learn was helpful in the implementation of dataset splitting into training and test sets, performing k-fold cross-validation, and evaluation of models using accuracy scores, precision, recall, f1 scores and confusion matrices [43]. Scikit-learn became one of the most efficient tools in optimising and comparing the models since it has a gentle learning curve and consistent interface that easily encourages play with various machine learning techniques [43].

Aspersions are standard algorithms like Random Forests, Support Vector Machines, K-Nearest Neighbours offered effortless assistance in providing a baseline to compare the results rendered by deep learning models like CNNs [43]. Because scikit-learn is easy to work with, concentrates on the implementation of important algorithms and contains descriptive references, it suited very well the experimental portion of this study and allowed models to be efficiently tested and validated within a fleeting time [43].

3.2.4 Scikit-image (skimage)

Scikit-image (skimage) is an extremely useful library for image processing in Python with many available algorithms for image processing tasks [42]. Its primary objective is to accomplish repetitive tasks like image resizing, image filtering, and image transformation which are crucial in preparing the image dataset for machine learning algorithms [42].

In this project, scikit-image played a key role in ensuring that all the input images were appropriately pre-processed, for example, by resizing and contract to the form and shape suitable for the neural networks [42]. Apart from shrinking, the other enhancement features available in the library include edge enhancement, histogram equalisation, etc. This improves the quality of the image data which is inputted in the

models and in return improves the performance of the models [42]. Due to its simple features and a lot of functionalities, scikit-image is an important software for working with image data in deep learning systems in a quick manner [42].

3.2.5 Pandas and NumPy

Pandas and NumPy indisputably belong to the most basic libraries aimed at working with data and performing numerical computations using Python programming language [11]. The frameworks such as Pandas are immensely popular for working with structured data sets, particularly textual ones like CSV and data frames crucial for subsequent data analysis and data preparation steps in machine learning processes [11]. This allows for the creation of intelligent and complex data models with easy-to-manage tabular data forms such as manipulation, filtering, and aggregation of data [11].

In this project, Pandas functions were used in loading and cleaning the datasets to make sure that the datasets were consistent and well-formatted and free of noise and other artefacts before feeding them into the different machine learning pipelines [11].

Alternatively, NumPy is the basic package in Python for performing numerical operations. It provides the representation of multi-dimensional matrices, which are key structures for deep learning that carry images and model parameters [11]. It was efficient in matrix manipulations and linear computation that made normal operations and transformations of large volumes of data faster [11].

In this project, quite a large dependency on NumPy was present when working with image data in arrays, considerably when normalising pixel values, and performing the needed calculations related to training the neural networks [11].

3.2.6 Matplotlib

Matplotlib is one of the most appreciated libraries amongst all, with respect to visualisation of data in Python [56]. It not only supports static and interactive graphics, but also provides facilities for animated graphics. Matplotlib was indeed useful in monitoring and presenting pivotal factors of the model training and performance in this project [56]. For instance, it has been used to create plots of training as well as validation accuracy, plot loss curve for training as well as validation, learning rate against the number of epochs etc, all such tools understanding how the model is interacting with the training process [56].

Additionally, Matplotlib was used to visualise results of data analysis, example images, data distribution and confusion matrices among others, which assisted in understanding the experimental results [56]. Thus, this application is especially useful during the project to provide quality and relevant graphical representation because of its adaptability and ample user manual [56].

3.2.7 Seaborn

Seaborn is a library for statistical graphics which is based on Matplotlib but is much more user-friendly than Matplotlib in the sense that it provides better looking and easier to interpret visualisations of quantitative data [45]. The creation and presentation of different graphs such as heat maps, box plots, and pair plots are made easy by the availability of superset functions [45].

Within the current project, Seaborn features were used to display and improve graphic visual interpretations of model evaluation and dataset related metrics. The beauty and simplicity by which trends and relationships in statistical data are presented further makes Seaborn well suited for those tasks [65].

To explain, Seaborn's heatmap was employed to show how specific variables correlated with model's performance, thus allowing deeper insights on how various participants impacted on the results [65]. Its clean and elegant interface, designed within a framework combining the enormously powerful panda's data structure, is superb for creation of high impact artificial intelligence visualisations [45].

3.3 Optimisation and Regularisation

Optimisation and regularisation are critical components in training deep learning models, as they directly affect the efficiency, convergence speed, and generalisation capability of the model [15]. The following sections describe the optimisation algorithms and regularisation techniques employed in this project to ensure optimal performance and prevent overfitting.

3.3.1 Adam Optimiser

Adam (Adaptive Moment Estimation) is regarded as the most advanced level of adaptive moment estimation which is basically an optimisation procedure utilised to speed up the training of deep learning architectures [20]. It calculates the adaptive learning rates for each parameter by keeping the first moment ('mean') & second moment ('uncentered variance') of the two moving averages [15]. This has enabled Adam to increase the rate of learning with respect to each weight in the networks & as a result reach faster training times, especially for deep networks with high data samples [15].

Unlike the simple gradient descent methods, Adam is defined in the sense that it scales the updates per parameter depending on the moment-based estimates of lower order which enables it to work better with sparse gradients and noisy data [20]. The most notable aspect of Adam optimisation is how the learning rate adapts while the model is trained [15]. Convolutional Neural Networks (CNNs) and other complex models

benefit from this since it provides robust and secure optimisation throughout the training epochs [20].

3.3.2 ReduceLROnPlateau Callbacks

These are familiar regularisation techniques that aim to reduce the problem of overfitting, as well as efficiently manage the training [54]. The issue of learning rate where the ReduceLROnPlateau is concerned is specifically for the learning phase [16]. The latter lowers the rate of learning via a coefficient when metrics like validation loss rates have achieved a plateau for an indeterminate length of epochs [54]. As a result, the learning rate is decreased to allow better and more thorough training of the model after the rapid increase [16]. In summary, incorporating Early Stopping and ReduceLROnPlateau into the training process will rather lead to enhanced efficiency while decreasing the risk of overfitting the model and thus, improving its generalisation capability [16].

3.4 Data Augmentation

One of the key features of data-scaling techniques is that it increases the number of available training datasets without necessarily going out to gather new datasets [3]. Instead, the approaches allow building more robust models which generalise much better and reduce the chances of overfitting by performing these manipulations to the already present images [3]. In this project on the other hand, techniques of data augmentation have been used to augment the variations of the CIFAR-100 data to better prepare the model to view objects differently [3].

3.4.1 ImageDataGenerator

The ImageDataGenerator module from Keras is a basic and crucial component to include, as it performs real-time data augmentation on the training dataset [36]. This module produces and reproduces batches of image datasets in assessment cycles with their combinations and random affine transformations such as rotating, zooming, shifting, flipping, and changing brightness [36]. With these augmentations, the model gets to experience more scenarios helping it perform well in new data after training.

These two include rotation as well as flipping whereby the model attempts to rotate the picture in efforts of trying to view the object from a different angle [36]. Coupled with zooming and shifting adds to the effectiveness of the model in the sense that the model can locate things even if the things are cropped out or resized [36]. ImageDataGenerator minimises the model's focus to a particular set of training data, enabling the model to be less sensitive to the conditions in the actual world by incorporating these operations [36].

3.5 System Utilities

The system utilities take up the most fundamental role of overseeing and controlling the environment where the development and training of machine learning models take place. These utilities provide direct file system access to handle files, upload data sets, or even save and retrieve models to ensure that the workflow is uninterrupted [64].

3.5.1 OS Module

OS module in Python is immensely helpful due to its ease of use in performing operations with an operating system [64]. Specifically, the OS module was used for reading, writing, or organising files within a given project like assigning paths to load datasets from, working with folders to store the models after training, as well as managing the outputs. Hence, there are no dependence issues in relation to any platform since the code compiled can run on any operating system [64].

Apart from managing files and their structure SRU also contains all features related to directories and since SRU deals with the automation of tasks so working with file directories is essential not only for simple file storage but also in relation to data preprocessing and model checkpointing during training phase [64].

3.5.2 Tarfile

The Tarfile module is a Python library that deals with compressed archives file formats especially .tar files which are used to store and transfer a large amount of data in a compact form [13]. In this system, the tarfile module was used for the purpose of performing the extraction of datasets that were provided in the format of tar archives and compressed, thus aiding the easy transport of large datasets such as CIFAR-100 [13].

The ability to compress files allows obtaining and working with large datasets as it is possible to download silos without previously unpacking them which saves time and makes everything much simpler [13]. The extraction and manipulation capabilities of these compressed files provided by Tarfile enabled effective storage and retrieval of datasets used for training and testing of models hence multitasking during the evaluations [13].

Chapter 4

CIFAR-100 Dataset

The dataset CIFAR-100 was generated by Alex Krizhevsky, Geoffrey Hinton, and other researchers from the University of Toronto for the aim of picture categorisation [31]. Its design has helped it gain popularity over time [9]. An improved version of the CIFAR-10 dataset, the CIFAR-100 dataset was developed especially to test and gauge the robustness of machine learning and deep learning techniques [9].

4.1 Dataset Characteristics

- Number of Classes: 100
- Number of Images: 60,000 images in total
    - Training Set: 50,000 images
        - Training images: 30000 images
        - Validation images: 20000 images
    - Test Set: 10,000 images
- Image Size: 32x32 pixels, in RGB format (3 channels: Red, Green, Blue)
- Image Format: Colour images (RGB) with small resolution (32x32)

4.2 Class Hierarchy

The tailoring of the classification system of CIFAR-100 dataset is done in a way that enables the allocation of each of the 100 classes of images into 20 super classes. In each superclass there are five subclasses [31]. For instance, in the superclass “Vehicles 1” the subclasses include the following: Bicycles, buses, motorcycles etc.

Example Super classes and Subclasses:

- Superclass: Aquatic mammals
    - Subclasses: beaver, dolphin, otter, seal, whale
- Superclass: Flowers
    - Subclasses: poppy, rose, sunflower, tulip, orchid

4.3 Challenges of CIFAR-100

This section describes the CIFAR-100 dataset challenges that are well suited for testing of high-performance and more advanced deep-learning systems [31]. Small Scale In terms of image size only 32x32 pixels, it becomes impossible to separate the even slightly different objects or features from each other [31].

Fine Grained Recognition Bifurcation of a single class into 100 classes many of which can be related draws challenges on object recognition. Less Sample Size When armed with 500 training images per class, connections are pretty small thus creating overfitting problems in deep models [9].

4.4 Data Splits

The CIFAR-100 dataset can be divided into two basic components.

4.4.1 Training Set

50,000 images are included in its components that are often applied when performing deep learning training [9]. These images are tagged so that the models can learn different pattern and features corresponding to each class [9]. This dataset is broken down into 30000 images in the training set and 20000 images in the validation set [9].

4.4.2 Test Set

Forming 10,000 images, this set is used to test the propositional model built by the in-built model [9]. At the training phase, the model does not see these images, thus, an evaluation of how well it will generalise without any bias is done [9].

Chapter 5

Methodology

This part of the thesis details the research method which has been employed in this project and it is divided into five main aspects: data loading, data preprocessing, model architecture, model training and evaluation. These phases, which are interrelated and contribute to the successful completion of the project, are also presented in an image to be more visual and structured.

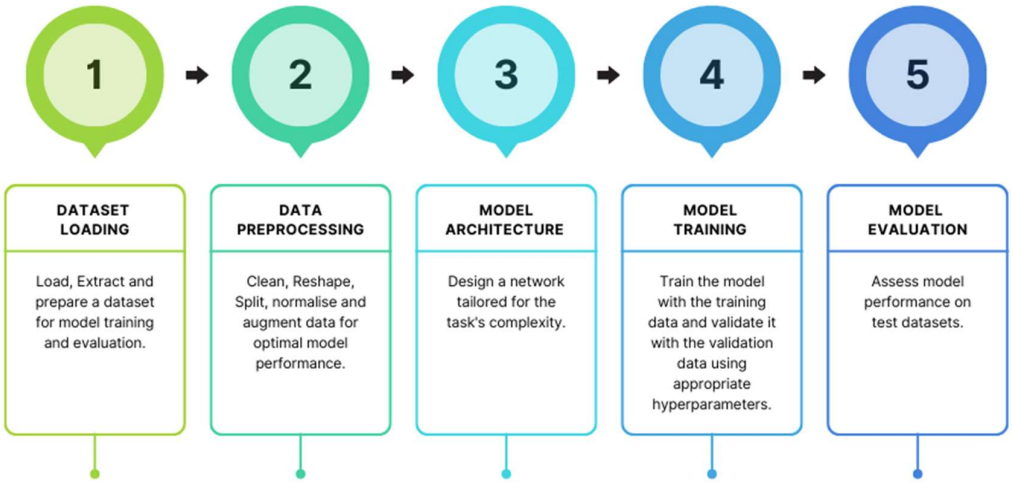

Image 5.1 - Stage of research method

5.1 Dataset Extraction and Loading

The CIFAR-100 dataset, which is well-known as a standard data set evaluated in image classification, was archived as a compressed '. tar.gz' file. This file format is designed to store various files and folders in a single file and therefore tends to come in handy when sharing large datasets. To make the dataset suitable for further operations, the contents were unpacked in a working directory using the Python tarfile module. This module saves a lot of time when working with '. tar.gz' form and helps in extracting the parts of the dataset without much hassle.

Upon extraction, the dataset was organised into three primary files:

- **Training data:** This file includes 50,000 images designated to one of 100 corresponding categories This file includes 50,000 images unified with distinct labels pertaining to the following one hundred object categories [9].

- **Test data:** The test data includes 10000 labelled images which are useful in determining how well the model has been trained. These images are essential for judging the capacity of the model to generalise since it has not been exposed to these images before [9].
- **Meta:** The file containing metadata highlights the names and the parent-subordinate relations of the coarser and the finer labels which explains how the information contained in the dataset is arranged the fine labels are made up of 100 granular-level constituents while the coarse labels are 20 hierarchical level broad categories [9]. This type of hierarchical labelling of the data facilitates more moderation in the examination of the data and even employs various levels of categorisation [9].

After extraction, the train, test, and meta files were unpickled and loaded into Python dictionaries. This step is crucial for converting the raw binary data into a usable format within Python, allowing for easier data manipulation and preparation in subsequent steps.

5.2 Data Preprocessing

Data preprocessing is one of the most important and initial steps in every machine learning or deep learning project [27]. It allows the incorporation of raw data being used into the most appropriate format for model training. Appropriate Preprocessing is very essential in the performance of machine learning models especially when one is handling complicated data like images [27].

In this project preprocessing the CIFAR-100 dataset was an important procedure which was required to use convolution neural networks (CNNs) with less complex architectures like EfficientNetB0 [30]. The pre-processing Pipelines consists of many important procedures which include data enrichment, data extraction, data augmentation, and data reshaping, data resizing all of which enhance the quality of the data and make it appropriate for model use [47]. Each of these steps is discussed in detail below and has been visually summarised in the image these are the narrative guidelines [47].

5.2.1 Data Reshaping

After the dataset is uploaded, the following stage constituted the alteration of the shape of the image data to suit the Convolutional Neural Networks (CNN). As a direct consequence, each of the images in the CIFAR-100 dataset is treated as a one-dimension array of 3072 values which defines the height, the width and the three colour channels [17].

Nevertheless, CNN involve architecture that is specific in relation to the input data format that is generally a four-dimensional array comprising such dimensions as

number of samples, height, width, and colour channels of the images [17]. Images were rescaled from their former shape of (num_samples, 3072) to (num_samples, 32, 32, 3) hence the desired shape using the powerful NumPy library. This reorients the information in a form that the CNN can utilise to be able to detect features in the images across the spatial extent of the images [17].

5.2.2 Label Encoding

The CIFAR-100 dataset also contains 100 fine labels, in addition to the 20 coarse labels that cover the former within the framework of the broader object classes. To understand and further feed in a classification model, it is crucial to convert the categorical labels onto some comprehensible and processor able form [10].

This was achieved using the to_categorical () function from the Keras library, which converts the fine labels into one-hot encoded vectors [10]. One-hot encoding transforms each label into a binary vector where only the index corresponding to the label's class is set to 1, and all other indices are set to 0. This encoding is critical for the output layer of the model to correctly interpret the predicted class probabilities during training [10].

5.2.3 Training and Validation Split

With these considerations in mind and in order to be able to determine how well exactly the model has been built, the dataset was divided into the two parts, training set and validation set. This split is fundamental to follow up how the model performs during training in a bid to prevent overfitting [50].

To perform this action, the StratifiedShuffleSplit method was used to split the data so that the distribution of classes was conserved in both the training and validation sets. If such stratification is not adhered to, then there would be class imbalance which would not be good for the model learning process [50]. A helpful 40 percent of the training data was used for the validation data set allowing for better tracking of how well the model generalises even as it learns [50].

5.2.4 Image Augmentation

To improve model reliability and avoid overfitting, the authors used data augmentation techniques to increase the diversity of the training dataset [26]. Here, some random changes were made to the input images by applying methods such as random cropping and horizontal flipping and changing the hues of the images [26].

These alterations create variations that are usually present in real life conditions such as angle, lighting or position and therefore make the model robust to such conditions [26]. To illustrate, Keras's ImageDataGenerator was employed in order to perform real-time augmentation during the training process so that any specific model was exposed to the new sets of the same data at every epoch [26].

5.2.5 Image Resizing

Certain models, such as EfficientNetB0, require input images of a larger size than the original 32x32 dimensions provided by CIFAR-100. To accommodate these models, each image was resized to 224x224x3 using OpenCV's resize () function [51]. This resizing is critical for ensuring that the input images match the required dimensions of the model, allowing the pre-trained architectures to effectively process the images and extract meaningful features [51].

5.2.6 Normalisation

Normalisation is a key preprocessing step that stabilises the training process by ensuring that all input features (pixel values) are on a similar scale [23]. Typically, image pixel values range from 0 to 255 [23]. These values were normalised to a range between 0 and 1 by dividing by 255 [23]. This transformation helps the model converge faster by reducing the risk of large gradients and improving numerical stability during training [23].

5.2.7 Visualisation

Such analysis was looked at and some images of the dataset were presented at various stages in the pipeline, to validate that the preprocessing has been carried out properly [32]. This included using a few selected images related to some labels and making up image sets in order to measure the effectiveness of additional techniques [32]. It became therefore evident that the images that were supposed to be pre-processed were indeed altered in the correct fashion and that the right stickers were placed [32]. This step also made it possible to view prospects of utilising data augmentation techniques to increase the diversity of the training set [32].

In this subsection, provision is made for elaboration of the novel data preparation system implemented considering all aspects of the dataset aimed at building complex neural networks. The use of cut, resize and relocate realistically adds value to the model especially when tackling the complex CIFAR-100 database [32].

5.3 Convolutional Neural Network (CNN) Model Architecture

This project involved the development of the custom Convolutional Neural Network Architecture to classify images from CIFAR – 100 datasets [14]. Aiming to further this goal, the CNN architecture is built in a way to progressively learn high level features of the input images with the purpose of distinguishing among 100 classes that exist in this dataset [14]. Below is a detailed explanation of the hardware and software components of the model outlining the architecture of the model layer by layer and relating the ways through which the model’s efficiency synthesis and generalisation are emblematically achieved [40].

5.3.1 Overall Structure

The sequential functionality of the architecture of the CNN is evidenced through the operation of the architecture in the sense that information out of a layer becomes a layer within another and hence, continues a linear presentation as however there exist many instances of cases where features are extracted and transformed [14]. The classical schema usually comprises four accumulative convolutional blocks with interspersed pooling and regularisation layers and several fully connected classification layers at the end to perform the classification task [40]. The last where the neurons for classification are employed is a multilayer perceptron and a SoftMax function transforms the outputs of the multilayer perceptron into probabilities to classify into 100 categories [14].

There are several strategies employed in the architecture to avoid what is commonly referred to as overfitting to the model, these strategies include use of overemphasized dropout, batch normalisation and others [40]. It also implies that they do not lose generalisation capability due to overfitting to the training set although the factors and the number of images in the CIFAR-100 set are rather few [14].

5.3.2 Convolutional Blocks

The core structure of the CNN is built within some few layers, representing mainly those layers which are convolutional blocks responsible for feature extraction using a filter on the input [14, 40]. Each block of CNN has been organised in such a manner so that as the block goes into deeper levels, it becomes capable of capturing higher levels and more abstract features from the data [14].

5.3.2.1. Two convolutional layers

Layers specifying a convolutional block indeed comprises of two ordered set of convolutional layers where every convolutional layer applies small's learnable patterns (3x3 kernels) which slide upon the image in search of other patterns, be it edges, surface texture, or pattern complexities [40]. These filters move mostly along the length and breadth of the image; hence they get to various positions and are in use at those positions [40]. In most of these layers the number of filters increase with increase in depth of the network, starting with 256 filters in first block and ending with 512 filters in other blocks. This increment aids the network to recognise finer and substantial number of features in the data as the layers deepens [40].

5.3.2.2. Batch normalisation

Batch normalisation is performed after every other convolutional layer. This procedure standardizes the output from the previous layer so that the distribution of data entering the next layer is stable and helps speed the training procedure [40]. It alleviates the

dependence on weight initialisations and prevents the network from getting trapped in local minima during optimisations [40].

5.3.2.3. ReLU activation

After performing the batch normalisation, the final non-linear transformation, which is a ReLU (Rectified Linear Unit) activation function is performed [25]. ReLU is an excellent choice since it is easy to implement and effective in assisting the network to learn patterns by providing a break to the linearity between the input and output [25].

5.3.2.4. MaxPooling

There is a MaxPooling layer which follows each convolutional block and that serves to decrease the spatial dimensions (height and width) of the feature maps by taking the highest value from every two-by-two area [25]. This down scaling procedure not only minimises the number of calculations within the network but also guarantees that the most salient characteristics are being looked for by the model making the network less susceptible to the small shift of the input images [25].

5.3.2.5. Dropout

To prevent overfitting, a dropout layer is added after the MaxPooling layer. During training, dropout randomly sets 30% of the activations to zero, effectively "dropping" them out of the network [25, 40]. This ensures that the network does not rely too heavily on any individual neuron, promoting a more robust and generalised learning process [25, 40].

1st Convolutional Block

- **Input Shape:** The model takes an image of (32, 32,3) size as the input which is 32 for height and 32 for width and three are for three colour RGB channels.
- **Convolutional Layers:** The first convolutional block comprises two Conv2D blocks, where each block is equipped with 256 filters and 3 (3 x 3) kernels. Batch Norm is also utilised after each Conv2D, followed by and without ReLU after the last one for Conv2D.
- **MaxPooling Layer:** A 2 by 2 MaxPooling layer reduces the spatial dimensions of the feature maps to increase the representation of the most notable features zones.
- **Dropout Layer:** 30% dropout is employed in the network to help in regularising it and avoid overfitting.

2nd to 4th Convolutional Blocks

These blocks follow the same structure and principles as the first block only increasing the number of filters to 512 filters per Conv2D. As for the elements of this convolutional network, the number of filter counts would always increase as these networks become deeper, thereby allowing the model to learn complex patterns and

intricate details in these images thus increasing the model's performance in discriminating the classes of the CIFAR-100 dataset.

5.3.3 Fully Connected Layers

The fully connected layers are only the final stage of the architecture as they do not introduce any further features but make use of those developed in the previous convolutional layers.

- **Flatten Layer:** The stack of 2D images known as the feature maps obtained after the last convolutional block is transformed into a 1D vector [40]. This transformation is carried out so that the spatial information may now be linear to be incorporated into the dense layers [40, 25].
- **Dense Layer:** Dense layers are followed by a 1000-unit dense layer which helps in learning different combinations of the features obtained from the convolutional layers [40]. The application of this dense layer gives the opportunity to the network to combine multiple features which enhances its performance in the classification problem [40, 25].

Learning is also restricted by the application of Batch Normalisation and Dropout after application of the dense layer to improve the quality of the model.

5.3.4 Output Layer

The purpose of the last layer of the Convolutional Neural Network is to produce the predicted class scores for the image samples.

- **Dense Layer with SoftMax Activation:** The output of this layer is 100 unit as there are 100 outputs corresponding to the 100 classes of CIFAR-100 data set. The activation unit can normalise the output so that they sum up to one, where each value denotes the chances of the image being categorised in that accurate class [40, 25].

Deep architectures plus the deployment of appropriate cut off mechanism such as dropout and batch normalisation make it possible to generalise when use the pretrained model from the CIFAR-100 data set [40, 25]. Because of this, the model that is developed will be able to extract features using a sequence of convolutional layers and perform classification using fully connected layers which is adequately flexible to be able to cater for complex images [40, 25].

5.4 DenseNet Model Architecture

For this project, a DenseNet model was carried out for the image classification task on the CIFAR-100 dataset [1]. DenseNet is also known as Densely Connected Convolutional Networks and it is an elaborate network which is meant to overcome the vanishing gradient problem, promote endeavours of providing input features again and assist in the forwarding of information from one layer to another [1]. An elaborated description of the DenseNet model which was used in this research is provided below:

5.4.1 DenseNet Overview

The basic principle of DenseNet structure is that every layer in the neural network must be feed forward connected to all the other layers [1]. All the layers take in the feature maps of the previous layers and all the layers give out the feature maps to the next layers [1]. This results in a finite connectivity structure, leads to less redundancy and improvement in parameter efficiency and makes it possible to have deeper networks with small parameters [1].

5.4.2 Model Parameters

- **Growth Rate:** The growth rate specifies gap controls the number of filters included in normalisation, training and building of any particular layer in a dense block. In this model, the growth rate is set to 24 [7].
- **Depth:** The dense net model consists of one hundred and sixty layers which includes convolutional layers, batch normalisation, ReLU and other layers structures incorporated in the model as well [7].
- **Compression:** compression factor of 0.5 is incorporated into the transition layers, which helps to cut down the number of feature maps by a half thereby contributing to the reduction of the size of the model [7].
- **Weight Decay:** L2 regularisation with a weight decay of $(1 * e{-}4)$ is applied to prevent overfitting by penalising large weights [7].
- **Classes:** The final layer outputs predictions for 100 classes, corresponding to the CIFAR-100 dataset [7].
- **Input Shape:** (32 * 32 * 3) is the image dimension of the ingested image which is also true for the images of CIFAR-100 [7].
- **Dropout Layer:** 50% dropout is applied to this neural network to assist in regularising the network to prevent overfitting [7].

5.4.3 DenseNet Architecture Components

The DenseNet architecture includes the following components:

5.4.3.1. Convolutional Layers

Every convolutional layer employ 3x3 filters to capture the spatial characteristics of the input image data [21]. The first convolutional layer generates a feature map with dimensions equating to 32 * 32 * number of channels.

5.4.3.2. Bottleneck Layers

A typical structure of a bottleneck layer contains a 1x1 convolution and a subsequent 3x3 convolution. A 1x1 convolution reduces the feature map count; consequently, new features can be extracted from the aggregated feature maps by applying the 3x3 convolution. This reduces the number of computations required without sacrificing the representational capacity of the network [7, 21].

5.4.3.3. Dense Blocks

It is finished completely connected by more than one layer inside the dense block [7]. Every layer in each block obtains input from all previous layers and sends output to all subsequent layers Thereby resource utilisation and descendants are treated rapidly improving the performance of the system by overcoming the downside of gradient sarcopenia [7, 21].

5.4.3.4. Concatenation

Within each dense block, the outputs of all previous layers are accumulated with the corresponding input prior to moving to the subsequent level [7]. This lets the model implement the features which have been incorporated in the earlier stages of modelling resulting in efficiency appropriate parameters usage [7, 21].

5.4.3.5. Transition Layers

Transition layers are placed between dense blocks. These layers perform two key operations:

- ***Compression:*** The transition layer defines the compression parameter and reduces size in terms of no of feature maps by this ratio (in this case it is 0.5) [7, 21].
- ***Down sampling:*** Average pooling of 2x2 filter faces and a filter step of two is used to downscale the feature maps. This helps to reduce the spatial size of the input, thereby improving the computational efficiency of the model [7, 21].

5.4.3.6. Batch Normalisation and ReLU Activation

Each layer has its output normalised by batch normalisation to improve the learning process and end up achieving faster convergence. To enable the model, to learn the

complex features from the input data, a ReLU activation block is added to the depths of the neural networks [7, 21].

5.4.3.7. Global Average Pooling

Once dense blocks and the transition layers have been traversed, global average pooling is performed to reduce each feature map to a single value, thus removing all but the spatial dimension [21]. This reduces the size information of the spatial information contained in the feature maps and converts it to a one-dimensional vector that can be input to the dense output layer [7, 21].

5.4.3.8. Fully Connected Layer (SoftMax Layer)

As the final layer, SoftMax activated fully connected (dense) layer is present. The layer performs class probability estimation for the 100 classes contained in sub-dataset where class probabilities are equally overwhelming given the input images [7, 21].

5.4.4 DenseNet Layer Calculation

The number of layers per dense block is calculated using the formula:

$$Number\ of\ Layers\ Per\ Block\ =\ \frac{Depth-4}{6} \qquad \text{-- 5.1}$$

In this model, the depth is 160.

$$Number\ of\ Layers\ Per\ Block\ =\ \frac{160-4}{6} \qquad \text{-- 5.2}$$

$$Number\ of\ Layers\ Per\ Block = 26 \qquad \text{-- 5.3}$$

So, each dense block holds 26 layers [1].

5.4.5 Summary of Model Architecture

The DenseNet construction comprises three intricate blocks being embedded in a becoming main body layer after each block completion. After the terminal dense block, batch normalisation and ReLU are carried out, after which comes the global average pool layer proceeded by the fully connected SoftMax output layer.

Finally, a model that incorporated the power of dense net of hundred categories of images from CIFAR-100 dataset has been created. As this architecture is effectively eager to feature reuse and maintain gradient flow throughout the network, it is likely to produce exactly accurate results.

5.5 EfficientNetB0-Based CNN Model

In this project and regarding the CIFAR-100 dataset, the EfficientNetB0 model was used as a base architecture for image classification tasks. EfficientNetB0 is also an established model in terms of Convolutional Neural Networks architectures which combines efficacy and economy in deploying resources making it appropriate for classification of many images. In this section, the model based on EfficientNetB0 architecture that was built in this work is explained in detail [2].

5.5.1 EfficientNet Overview

The EfficientNetB0 model belongs to the EfficientNet model series that uniformly expands all depth, width, and resolution parameters as a complete strategy. EfficientNetB0 is the baseline model which is having the smallest size and fast but maintains some reasonable accuracy. In this work, this model was initially trained on the ImageNet dataset and retrained on CIFAR-100 dataset [2].

5.5.2 Model Architecture

The pre-trained EfficientNetB0 model is used as the basic structure of the new model, which is then extended to contain additional layers so that it fits the CIFAR-100 dataset. The following are the functional parts of the architecture:

5.5.2.1. Pre-trained EfficientNetB0 Base

The EfficientNetB0 model was loaded without its top layers. This base model retains the weighted parameters obtained from ImageNet and therefore can extract high quality and transferrable features from images [2]. The input for the base model was defined as (224 * 224 * 3), which was based on the dimension of the input image. The feature extraction is done by the EfficientNetB0 model, which is a pretrained model, and all its convolutional layers are present [2].

5.5.2.2. Global Average Pooling Layer

Subsequent to this, a Global Average Pooling (GAP) layer is applied in the end to minimise the dimensionality of the feature map. In place of such fully connected layers, GAP operates by concentrating the content of the feature maps and reducing its spatial dimensions to one numerical central tendency of that feature map. This layer is helpful when decreasing the number of parameters required without eliminating the topological information of the feature maps [37, 2].

5.5.2.3. Dropout Layer

A Dropout layer with a rate of 0.3 (i.e., 30%) is added after the Global Average Pooling layer. Dropout is a regularisation technique that randomly sets a fraction of input units

to zero during training, reducing the risk of overfitting and improving the generalisation of the model [25, 40].

5.5.2.4. Dense Output Layer

The final layer consists of 100 units of fully connected Dense layer with each unit corresponding to one of the 100 labels/classes present in the CIFAR-100 dataset. The activation function used is 'sigmoid,' which is often applied for multi-class classification. The Sigmoid activation manages the network such that every class is in a way taken as a separate entity such that it drives the output of the classes to be in distribution [37, 2].

5.5.3 Input Shape and Number of Classes

- **Input Shape:** The input images have dimensions of (224 * 224 * 3), where the height and width of each image are 224 pixels, and the number of channels is 3, standing for the RGB colour space. These images are resized from the original (32 * 32) CIFAR-100 images to match the input requirements of EfficientNetB0 [37, 2].
- **Number of Classes:** The number of output classes is 100, matching the CIFAR-100 dataset, which holds 100 distinct object categories [37, 2].

5.5.4 Model Summary

The EfficientNetB0-based model was built using the Sequential API in Keras, and the architecture is summarised as follows:

The EfficientNetB0 is a base feature extractor that has been trained on ImageNet beforehand. After that, a global average pooling operation is performed. This operation condenses the feature maps of each channel into a single vector, removing all spatial dimensions [37, 2]. During training, to 'regularise' the model, dropout allows the selection of 30% of the neurons to be ignored and incorporated into the model. Finally, using the sigmoid activation, the dense layer of output is used to obtain the final predictions for 100 classes.

5.6 Model Training

The training phase is crucial in the development of the model as it requires the fine-tuning of several parameters, the incorporation of data augmentation approaches for better model generalisation, and the management of certain optimisations so that overfitting would be avoided as well as the best models be preserved [28]. This section describes in detail how the training phase was conducted particularly in terms of the key training settings and, the use of data augmentation techniques applying model checkpointing measures [28].

5.6.1 Training Configuration

5.6.1.1 Batch Size

Multiple batch sizes are used, and the model was trained with the chosen sizes that have been seen appropriate based on the empirical results and hardware. In training neural networks, batch size is the number of training examples utilised in one iteration, forward pass and people backward pass through the network [14].

In order to have faster convergence, small batch sizes such as 32 and 64 were adopted to make a greater number of updates to the weights making the convergence of the model smoother [14]. Smaller batches can add noise to the gradient estimates but are more capable of effective loss surface exploration which helps in evasion of flat regions of the basins [14]. Larger batch sizes of 128 and 256 were used to enhance operational speed but there is the disadvantage that more memory is used and there are fewer updates within an epoch [14]. Hence balance between the quickness of convergence and the cost of computation, was maintained while training [14].

5.6.1.2 Epochs

An epoch is other words is referred to as the number of times the whole visiting one training dataset is applied in the model. The model was eventually trained on an increasing number of epochs according to the results and the convergence [14].

The text explains the concepts of underfitting and overfitting in machine learning. It emphasises the importance of finding the right number of training epochs to ensure that the model learns from the data without overfitting [14]. It suggests training models in the range of 15 to 50 epochs [14].

5.6.1.3 Validation Split

While training, a validation split was employed to determine the remaining to assess the model performance on new data at the end of every epoch [6]. Forty percent of the training data was put aside for use as validation. The validation set was able to avoid overfitting or underfitting by tracking metric performance like validation loss and accuracy. Additionally, this data informed the learning rate adjustments [6].

5.6.2 Data Augmentation

To improve the generalisation of the model and its real-life usefulness, the authors employed several data augmentation methods in this line of work [3]. This is no more than what is generally referred to as increasing the existing test sets by adding supplemental datasets so that the training datasets become enriched [3]. This would imply that the model would incorporate all the major factors within a particular class and not subtle changes in the data leading to overfitting [3].

5.6.2.1. Random Flips

Horizontal flipping was applied to a random subset of the images. This operation reflects the images horizontally, enabling the model to learn from different orientations of objects [53]. This technique is especially useful in datasets like CIFAR-100, where the orientation of objects, such as animals or vehicles, can vary significantly. It becomes more robust to these transformations in the test set by exposing the model to mirrored versions of the data [53].

5.6.2.2. Random Rotations and Shifts

Images were subject to slight angular position variations or small positional variations in the directions of motion for slight angular or positional changes [44]. These transformations induce the model to become invariant to minor changes caused by repositioning and orientation of the objects, thereby enhancing its specificity in object recognition when these are not necessarily at the centre or straight within the frame [44].

5.6.3 Model Checkpointing

To optimise the training process and prevent overfitting, model checkpointing techniques were employed [60]. Implementing model checkpointing, the model's state was considered saved during the training when validation accuracy was the highest [60]. In every epoch, if the validation accuracy went up, the model was written on the hard drive to maintain the optimum version of the model [64]. This made it possible even recovering the best model can be used when the training was stopped or when the model performance has reduced at later training epochs because of overfitting [64]. This mechanism of checkpointing was extremely critical particularly to avoid an instance where the final model to be evaluated with is the worst version that has been trained [64].

5.6.4 Regularisation Techniques

To lessen the danger of overfitting and improve the model's effectiveness in handling a new example data, several regularisation techniques were applied during training. These techniques are necessary so that the model does not learn where certain patterns in the training data are too well and performs poorly in the test data [4].

- **Weight Decay (L2 Regularisation)**

L2 regularisation or weight decay techniques were also used in the Dense layers of the model [4]. In this method, a cost term is introduced in the loss function which includes the squared value of weights associated with the model [4]. Since excessive weight values are not encouraged, weight decay seeks to ensure that the network locates appropriate, but more importantly, simpler, and more efficient solutions which foster less overfitting than would often be the case [4]. This reduces the chances of overfitting since the model cannot ‘overfocus’ on any given input feature, which reduces any

noise from the training set. Typically, L2 regularisation influences rather smoother learned weight distributions so that the developed models are robust to several datasets [4].

5.6.5 Learning Rate Scheduler

During the training period, it was decided that learning rate scheduling should be used. When the validation loss has plateaued out this is what they call reaching a plateaued stage of retraining therefore it is time to lower the learning rate [52]. In such a way, it is beneficial for the model to adjust the learning rate because the model is already close to the optimum and requires only slight changes [52].

A learning rate "scheduler" was incorporated during the entire course of training to alter the working rate whenever the circumstance called for it [52]. The implementation of this method improves the model by effecting changes of lesser magnitudes as the model approaches convergence. To prevent clogging of the training, the ReduceLROnPlateau callback also contributed when the validation loss seemed constant so that the rate of learning was reduced [52].

5.7 Model Evaluation

Following the completion of the training phase, a model evaluation exercise was carried out on test set containing 10,000 images to determine how well the model performs. The evaluate function was used in these proceedings to determine the test loss and accuracy metrics which help in understanding the model performance on data that it has not been exposed to in the previous stages [12].

After training, the model was evaluated with a test set from the CIFAR-100 containing 10,000 images [12]. From Keras the evaluate function was applied to estimate the model’s loss and accuracy for the test data, information that has been significant in assessing model performance on data that was not trained on [12]. The evaluation metrics for model comparison included the impact of different regularisation methods and data augmentation on accuracy and robustness. Profile analysis of the training/validation performance vs testing performance was used to determine that the structure achieves optimal model performance [12].

Chapter 6

Majority Theory

Majority voting is one of the most common techniques used in ensemble learning which helps to increase the accuracy and robustness of the results by amalgamating the predictions of several classifiers [61]. In a system that employs majority voting, the final prediction arrives at the class accepted by the majority of the models (classifiers) in the ensemble of models, based on their votes [58].

Other metric strategies include error rates such as False Positive Rate (FPR) and False Negative Rate (FNR), which can be used to categorise the various kinds of errors that a specific model is likely to make [61, 62, 58]. For this reason, in this project, more focus is directed toward the basic machine learning models such as EfficientNet, DenseNet and the CNN architectures for object recognition applications that require more exact adjustment of the above-stated error rates.

Collective or majority cognitive responses provide a decision-making structure that can be practically employed in algorithm-based decision-making where multiple models are combined, and their outputs are aggregated [62]. Aggregation of empirical prediction of a decision theory is the focus of majority theory because better decision-making accuracy is assumed after making several predictions previously [58]. This report will use majority theory, False Positive and Negative results to help in the Choice and evaluation of the performance of the Convolutional Neural Networks (CNNs) [61, 58].

6.1 Understanding False Positive and False Negative Results

False Positives (FP) ambiguously examines a negative instance compared to the case non-finding any one positive criteria in a case [61, 58]. Immediately, it is straightforward to see that FN occurs as an outcome of missing diagnosing that which is presumably positive [58]. Diagnostic errors of this kind are bad no matter the applications here. Still, people will be particularly careful about the FNs and FPs when it comes to certain high-risk contexts like medical diagnostics or self-driving cars where any of them might spell great horror or negative outcomes [61, 58].

For example, in case of a medical niche, false negatives (i.e. a missed diagnosis of a disease) can lead to the missed opportunity of curing the disease, while false positives can cause excessive worry or inappropriate treatment [34]. It brings us to the conclusion that compared to the other models that the prediction should not only be

reflected on how well the model performs overall but also on the models' FPR and FNR ratios [34].

6.2 Majority Theory and Algorithm Decision-Making

Majority theory is understood based on collective decision making whereby a single decision is taken with the 'suffrage' of several classifiers. When working with several convolutional neural networks each of such networks would tend to have certain areas of strengths or weaknesses [62]. However, when their individual contributions are combined, there are instances where the chances of committing a fault either false positive or false negative come down, thus improving the coalesced result [62].

- **How Majority Theory Works:** Suppose we have three models - EfficientNet, DenseNet, and a traditional CNN - working on the same classification problem, and each model provides a prediction [62].
- **Vote Aggregation:** For a given instance, if two out of the three models predict a certain class, the final decision will reflect the majority vote [62].
- **Weighting the Models:** In some cases, models that prove lower FPR or FNR may be given more weight in the decision-making process, making them more influential in the majority vote [62].

Majority theory helps mitigate the impact of the weaker model's inaccuracies, thereby improving the robustness of the final prediction by aggregating decisions from different models.

6.3 Decision Criteria Based on FPR and FNR

In classification issues, the trade-off between FPR and FNR is usually figured out by the nature of the problem [5]. As an illustration, take a security surveillance system, it may be more tolerable to have a higher proportion of false positives than false negative rates, for instance, while most threats are imposed, they are not wholly identified more threats are assumed instead of having false negatives where an actual threat is present but is not identified [5]. As in most cases in medical diagnosis, people are generally more inclined towards the reduction in false negatives in order not to miss any illness that requires treatment.

The decision criteria based on FPR and FNR can be summarised as follows:

- **Low FPR Focus:** Models are prioritised when the risk of false positives incurred is substantial (for instance in the criminal justice systems where there should be no wrongful accusations) [5].
- **Low FNR Focus:** Models are prioritised when the risk of false negatives incurred is substantial (Such as in healthcare where a loss in diagnosis could have very severe outcomes) [5].

- **Balanced Approach:** This will apply for such situations where inhibition of FPR and FNR is as true importance to perfecting either or both will be needed, in this case using a weighted best of FPR and FNR may be necessary [5].

In deciding which model to prioritise on the application based on FPR and FNR, EfficientNet, DenseNet and traditional CNN models FPR and FNR will be used on a CIFAR-100 dataset [5].

For instance, if DenseNet exhibits a lower FNR but slightly higher FPR than EfficientNet, it might be chosen for medical applications, while EfficientNet, with its balance of both rates, could be more suitable for general-purpose tasks.

Chapter 7

Results and Analysis

The ultimate step of the process included training the CNN on the CIFAR-100 dataset and recording the metrics to evaluate the model developed. In this section, the results are further elaborated, beginning with the accuracy, loss, and a majority theory, as well as the extent of overfitting that the model experienced when applied to unseen data. The analyses also captured the difficulties experienced during the training and possible avenues for improvement.

7.1 Model Performance

All models underwent the processes of training and validating for image classification on the CIFAR-100 database during the training, validation, and test stages. The factors measure the performance and the extent to which each model can generalise to new data and the actual areas that require improvement.

7.1.1 Training Accuracy

The model achieved a prominent level of training accuracy which implied that the model was capable of learning and accurately modelling the training data. This shaped pattern was seen towards at the end of every epoch making it more convincing that the model managed to learn the data.

- **CNN Training Accuracy:** The Convolutional Neural Network (CNN) was able to reach approximately 99.38% in terms of training accuracy.
- **DenseNet Training Accuracy:** As for training accuracy, the gloomy deep learning, and the dense net architecture reached the approximate value of 99.74%.
- **EfficientNet Training Accuracy:** The training accuracy of this model structure, EfficientNetB0 model structure, was around 91.5%.

7.1.2 Validation Accuracy

The other performance measure, validation accuracy, measured the model effectiveness on the validation subset of the training data was also quite effective. There was further a limitation on how many epochs an iteration could take with regards to the validation accuracy of the model. This made sure that although the model performed well on training data, it was not too specifically trained for such data by continuing to train the model when validation accuracy had saturation.

- **CNN Validation Accuracy:** The validation accuracy of the convolutional neural network (CNN) was around 61%.
- **DenseNet Validation Accuracy:** DenseNet achieved a validation accuracy of around 64.5%.
- **EfficientNet Validation Accuracy:** EfficientNetB0's validation accuracy reached around 81%.

7.1.3 Test Accuracy

On the CIFAR-100 test set (10,000 images), the model achieved a respectable accuracy, proving its ability to generalise to new, unseen images. The test accuracy provides the most realistic estimate of the model's performance in real-world scenarios.

- **CNN Test Accuracy:** On the CIFAR-100 test set, the CNN model achieved a test accuracy of around 61%.
- **DenseNet Test Accuracy:** The DenseNet architecture achieved a test accuracy of around 64.66% on the dataset.
- **EfficientNet Test Accuracy:** The EfficientNetB0 model outperformed the other architectures with a test accuracy of around 81%.

7.1.4 Summary

The model demonstrated a consistent decrease in train loss, proving its capability to minimise the prediction error, while validation loss was still comparatively higher, which was expected. Upon observing the confusion matrix results, the model performed satisfactorily in predicting distinct classes (for instance vehicles and animals) and not visualised similar types (which contain types of trees or objects) [5]. Certain classes yielded weaker performance and commensurate accuracy because of dataset built-in difficulty [5].

In this process, it was set up that the performance level that could be derived from the base classifiers increased upon use of majority voting [5]. This ensemble method contributed to performance homogeneity across classes of very divergent nature, and lessening some of the model's limitations for the difficult-to-classify categories.

Precision, recall, the F1-score supplementary to deciding more rationally the provided class metrics above was also being assessed in this case [5]. It was observed however that when it came down to one edge of the precision and recall curve, there was a low rate of false positives, which is an extremely healthy thing in such areas as medical diagnosis [5]. It was noted that high recall depicted aggressiveness in acquiring positive instances, while the F1-score was used to assess the balance between recall and precision [5]. Support metrics were used to illustrate the number of actual instances that were correctly classified into the designated per class [5].

7.2 Results based on accuracy

The results of the investigation demonstrate the best performance of the EfficientNetB0 model compared with conventional CNN architectures both in accuracy and computation cost [5].

The argument here is that, unlike standard CNNs, EfficientNetB0 does not just expand its network along edges, but rather networks feature indiscriminately thereby enhancing both feature extraction processes and classification accuracy with respectful constraint on computation resources [5]. The model was found to be more or equally effective than all the methods on evaluation metrics as the study was undertaken towards offering solutions to image classification tasks that are relevant in most cases in the real world [5]. These results point to the advantages of EfficientNetB0, which is a well-balanced architecture where more than network parameters can be tuned at the same time to achieve high accuracy in predictions.

7.3 Comparison and Interpretation of FNR and FPR

1. False Negative Rate (FNR) Analysis

- **EfficientNet:** Notable for the fact that it recorded the best FNR of 19.5%. This means that instances labelled as positive are done so correctly most of the time. Such applications often entail positive instance errors (false negatives) that could be quite undesirable and even lethal like medical diagnosis and security systems.
- **DenseNet:** Followed with an FNR of 35.5, which was an improvement over CNN, indicating that its ability to find positive instances was higher but not as high as EfficientNet.
- **CNN:** Annotated an FNR rate of 39%, thus, ranking lowest in the three models in terms of detecting positive instances.

2. False Positive Rate (FPR) Analysis

- **EfficientNet:** For the last time, the model is forward-looking based on user role with an FPR of 0.19% and thus caused the least false alarms, (negative instances wrongly classed as positive) Incorrect Positive Rate or Positive Predictive Value.
- **DenseNet:** As such, A slightly higher incidence of false positives was expected, and a higher incidence of 0.35% was observed. The level of false positives was increased when compared to EfficientNet.
- **CNN:** Recorded the highest FPR at 0.39%, indicating it produces the falsest positives among the three models. Although the differences in FPR are smaller than in FNR, EfficientNet still proves to be the most reliable model in minimising false positives.

7.4 Decision-Making Using Majority Theory

These models give us the FNR and FPR for each of the models. Hence, this study is permitted to use majority theory to optimise the process of planning on which model to take or the output of which models to combine for the final decision.

When comparing the efficiency of models, respectively, EfficientNet is more effective than CNN and DenseNet in both FNR and FPR. The model achieves success in identifying the greatest number of positive instances (low FNR) and reducing false positives (low FPR), thus makes it the most effective model. Therefore, in scenarios where both errors have profound consequences, the use of EfficientNet would be most appropriate.

- **Majority Voting Approach**

These models were able to get FNR and FPR for all the models. Hence, in this study, there is an advantage of utilising the majority theory in the selection of what model to use and which output of the models to queue up for the final decision-making process.

For instance, when two or more models predict an A and one model B, the final output would be in favour of A. Since the performance of the EfficientNet is the most favourable, then the number of votes to be raised would be in which the majority opinion chooses the right definition.

The highest overall accuracy achieved on allowing a vote by the models received 52.09% giving respect to the majority voting criteria. It remained stable around the achieved level of 67.06% after having a weighted majority vote on the same competition. In this study, the majority assumed an accuracy level of 62.25% with reinforcement voting using three different weighted votes for the majority 25% DenseNet 25% CNN and 50% EfficientNet.

Since EfficientNet achieved the highest accuracy, the goal of this research was to achieve the maximum accuracy realising that 50% of the weights would be distributed to EfficientNet. This trade-off improved the cross-validation score on all the output classes although a corresponding decrease in accuracy was observed.

7.5 Decision, Performance and Scalability

The research findings demonstrate that EfficientNet shows the lowest number of false negatives and false positives, indicating its readiness for deployment with minimal risk of inaccuracies. The study utilised EfficientNetB0 with compound scaling for higher accuracy and fewer computational resources, and implemented measures to prevent overfitting, such as ImageDataGenerator, dropout layers, and advanced data augmentation techniques.

The majority voting technique combined predictions from CNN, DenseNet, and EfficientNetB0 to enhance system resilience. While EfficientNetB0 alone had the highest accuracy, majority voting improved prediction consistency across a wider range of classes. Weighted majority voting with 50% weight for EfficientNetB0 and 25% each for CNN and DenseNet resulted in significant improvement in overall accuracy and robustness, reducing the likelihood of misclassifications.

EfficientNetB0 and majority voting offers benefits for real-world applications, including higher accuracy, lower computational costs, and suitability for diverse applications such as medical diagnostics, retail automation, and autonomous vehicles.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The research findings demonstrated the effectiveness of utilising the EfficientNetB0 architecture for image classification tasks, particularly when applied to the CIFAR-100 dataset. The model exhibited commendable accuracy and efficiency in handling the challenge of categorising images into 100 distinct classes. This success has been made possible with the use of transfer learning, which compressed the previously divergent tasks into a single task, thus resulting in faster improvement than ordinary.

As well, the model in this instance benefited from the application of regularisation strategies such as dropout or L2 weight decay which protected the model from being overfit. Employing ImageDataGenerator technique has also been helpful in avoiding overfitting and improving the model's abilities to make predictions on test data. Despite encountering limitations stemming from the restricted volume of data and the relatively low image resolution inherent in the CIFAR-100 dataset (32 by 32 pixels), the study yielded positive results.

One of the most interesting components of this work was using the majority theory since it enabled the voting of results of various models which include CNN, DenseNet and EfficientNet. Such an ensemble led us to obtain an accuracy of 67% of the CIFAR-100 test set through majority voting using a same weight for every model. This way enabled every weakness and strength of a particular model to remain intact thus increasing the accuracy and the robustness of the ensemble.

The research has effectively established a solid benchmark for future endeavours in this area, underscoring the suitability of EfficientNetB0 for dealing with intricate tasks involving small datasets. It was emphasised that achieving optimal performance necessitates the application of advanced techniques, including data augmentation, and learning rate scheduling, for fine-tuning the model. These findings offer promise for further advancements in the performance of the EfficientNet family and its architectural variations in upcoming studies.

8.2 Future Work

Despite the best performance achieved free from the EfficientNetB0 model, there still exists a wide scope for further improvement and experimentation. In future work, this research could be further developed in various directions:

8.2.1 Increasing Training Duration

Improvement of up to 10% in the absolute accuracy of the model could be achieved by increasing the training epochs of the model. The above allows an allowance for higher epochs when large datasets are being trained or when performance barriers of the model are being looked for.

8.2.2 Investigating Other Advanced Architectures

Modern architectures like DenseNet, ResNet, Vision Transformer among others aside from EfficientNet can also still be applied. Such structures have been well adopted in lots of image classification tasks and may even perform better than Efficient Net especially when they are professionally trained on the dataset.

Furthermore, the research may be extended towards more effective members of the efficient net such as efficient net B1, B2 and above Efficient nets. Although these larger structures have an equal increase in the level of intricacy, they are however built with more layers and as such, should be able to pick up more information about images.

8.2.3 Improving Image Resolution

It will be acknowledged that another option that can be pursued at a future date is that of using augmented input images in place of original images. With regards to the CIFAR-100, however, this is not the case because it has tiny images that are 32x32 pixels. In future with bigger image datasets having straining models for targeting computer machine graphics more general composited would be easier.

8.2.4 Expanding the Dataset

Another appropriate model development strategy, which would improve the generalisation of the network, entails increasing the existing dataset, for example, through the use of synthetic data generation techniques such as GANs or techniques of data augmentation. This model approach adds more features to the model and thus improves the performance of the model when applied to the CIFAR-100 dataset and any other image classification tasks.

8.2.5 use of ImageDataGenerator

To avoid overfitting, applying the ImageDataGenerator may help since it offers a variety of procedures for example data augmentation, normalisation, and other forms of preprocessing that aid in reducing the overfitting of machine learning systems that are image-based. These techniques assist in developing a more comprehensive and

generalised model by offsetting the disadvantages posed by overfitting due to the use of modified versions of training images.

The model could also be further pushed to seek the extent of performance that could be achieved when using considerably smaller and badly designed datasets. If the level of sophistication increases, the model would expand methods of overcoming issues and improving the current accuracy and robustness on the classification task so that wider order impact on imaging classification on its practical occurrence would be possible.

Bibliography

1. Alejandro Ito Aramendia (2024). DenseNet : a Complete Guide - Alejandro Ito Aramendia - Medium. [online] Medium. Available at: https://medium.com/@alejandro.itoaramendia/densenet-a-complete-guide-84fedef21dcc [Accessed 19 Sep. 2024].
2. Alhijaj, J.A. and Khudeyer, R.S. (2023). Integration of EfficientNetB0 and Machine Learning for Fingerprint Classification. Informatica, 47(5). doi:https://doi.org/10.31449/inf.v47i5.4724.
3. Amazon Web Services, Inc. (n.d.). What Is Data Augmentation? - Data Augmentation Techniques Explained - AWS. [online] Available at: https://aws.amazon.com/what-is/data-augmentation/#:~:text=Data%20augmentation%20is%20the%20process.
4. Another Deep-Learning Blog. (2020). Understanding L2 regularization, Weight Decay and AdamW. [online] Available at: https://benihime91.github.io/blog/machinelearning/deeplearning/python3.x/tensorflow2.x/2020/10/08/adamW.html.
5. Awoyele, T. (2020). Confusion Matrix, Precision , Recall and F1-Score. [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/confusion-matrix-precision-recall-and-f1-score-d5f340e38cca.
6. Baheti, P. (2021). Train, Validation, and Test Set: How to Split Your Machine Learning Data. [online] V7labs.com. Available at: https://www.v7labs.com/blog/train-validation-test-set.
7. Baldha, S. (2022). Introduction to DenseNets (Dense CNN). [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/03/introduction-to-densenets-dense-cnn/.
8. Bangar, S. (2022). LeNet 5 Architecture Explained. [online] Medium. Available at: https://medium.com/@siddheshb008/lenet-5-architecture-explained-3b559cb2d515.
9. Beniar, B. (2022). BillyBSig/CIFAR-100-TFDS. [online] GitHub. Available at: https://github.com/BillyBSig/CIFAR-100-TFDS.
10. chugh, aakarsha (2018). ML | Label Encoding of Datasets in Python. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/.
11. Codecademy. (n.d.). Introduction to Pandas and NumPy. [online] Available at: https://www.codecademy.com/article/introduction-to-numpy-and-pandas.
12. domino.ai. (n.d.). What Is Model Evaluation? | Domino Data Science Dictionary. [online] Available at: https://domino.ai/data-science-dictionary/model-evaluation.

13. dotnet-bot (2024). TarFile Class (System.Formats.Tar). [online] Microsoft.com. Available at: https://learn.microsoft.com/en-us/dotnet/api/system.formats.tar.tarfile?view=net-8.0 [Accessed 19 Sep. 2024].
14. Dutta, S. (2024). Designing Your Own Convolutional Neural Network (CNN) Model: a Step-by-Step Guide for Beginners. [online] Medium. Available at: https://medium.com/@sanjay_dutta/designing-your-own-convolutional-neural-network-cnn-model-a-step-by-step-guide-for-beginners-4e8b57836c81.
15. Elshamy, R., Abu-Elnasr, O., Elhoseny, M. and Elmougy, S. (2023). Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning. Scientific Reports, [online] 13(1), p.8814. doi:https://doi.org/10.1038/s41598-023-35663-x.
16. fastai. (2024). Tracking Callbacks – Fastai. [online] Available at: https://docs.fast.ai/callback.tracker.html [Accessed 19 Sep. 2024].
17. FutureLearn. (2022). How Do You Reshape a Data set? [online] Available at: https://www.futurelearn.com/info/courses/introduction-to-data-analytics-with-python/0/steps/262808#:~:text=The%20shape%20of%20a%20data [Accessed 20 Sep. 2024].
18. GeeksforGeeks (2024a). CIFAR 100 Dataset. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/cifar-100-dataset/ [Accessed 19 Sep. 2024].
19. GeeksforGeeks (2024b). Efficientnet Architecture. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/efficientnet-architecture/#:~:text=Le%20from%20Google%20Research%20in [Accessed 19 Sep. 2024].
20. GeeksforGeeks. (2020). What Is Adam Optimizer? [online] Available at: https://www.geeksforgeeks.org/adam-optimizer/.
21. GeeksforGeeks. (2024c). DenseNet Explained. [online] Available at: https://www.geeksforgeeks.org/densenet-explained/.
22. GFG (2024). What Is Object Detection in Computer Vision? [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/what-is-object-detection-in-computer-vision/.
23. Gopal, S., Patro, K. and Kumar, K. (n.d.). Normalization: a Preprocessing Stage. [online] Available at: https://arxiv.org/pdf/1503.06462.
24. Guide, U. (2024). api4ai. [online] api4ai. Available at: https://api4.ai/blog/object-detection-for-images-technologies-applications-and-best-solutions [Accessed 19 Sep. 2024].
25. Gurucharan, M. (2020). Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network. [online] upGrad blog. Available at: https://www.upgrad.com/blog/basic-cnn-architecture/.
26. Iceland, M. and Kanan, C. (2023). Understanding the Benefits of Image Augmentations. [online] Available at: https://arxiv.org/pdf/2306.06254 [Accessed 4 Jun. 2024].

27. Idan Novogroder (2024). Data Preprocessing in Machine Learning: Steps & Best Practices. [online] Git for Data - lakeFS. Available at: https://lakefs.io/blog/data-preprocessing-in-machine-learning/#:~:text=Data%20preprocessing%20is%20critical%20in [Accessed 20 Sep. 2024].
28. Iguazio. (n.d.). What Is Model Training. [online] Available at: https://www.iguazio.com/glossary/model-training/.
29. Jason Brownlee (2019). A Gentle Introduction to Object Recognition with Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/object-recognition-with-deep-learning/.
30. Khanna, C. (2021). CIFAR 100: Transfer Learning Using EfficientNet. [online] Medium. Available at: https://towardsdatascience.com/cifar-100-transfer-learning-using-efficientnet-ed3ed7b89af2.
31. Krizhevsky, A. (2009). CIFAR-10 and CIFAR-100 Datasets. [online] Toronto.edu. Available at: https://www.cs.toronto.edu/~kriz/cifar.html.
32. Llaha, O. and Aliu, A. (n.d.). Application of Data Visualization and Machine Learning Algorithms for Better Decision Making. [online] Available at: https://ceur-ws.org/Vol-3402/short01.pdf [Accessed 20 Sep. 2024].
33. Luz, E., Silva, P., Silva, R., Silva, L., Guimarães, J., Miozzo, G., Moreira, G. and Menotti, D. (2021). Towards an effective and efficient deep learning model for COVID-19 patterns detection in X-ray images. Research on Biomedical Engineering. doi:https://doi.org/10.1007/s42600-021-00151-6.
34. manoa.hawaii.edu. (n.d.). Practices of Science: False Positives and False Negatives | manoa.hawaii.edu/ExploringOurFluidEarth. [online] Available at: https://manoa.hawaii.edu/exploringourfluidearth/chemical/matter/properties-matter/practices-science-false-positives-and-false-negatives#:~:text=The%20patient%20may%20be%20diagnosed.
35. Mathworks.com. (2019). Object Recognition. [online] Available at: https://www.mathworks.com/solutions/image-video-processing/object-recognition.html.
36. Maximinusjoshus (2021). Image Data Augmentation Using Keras ImageDataGenerator. [online] Featurepreneur. Available at: https://medium.com/featurepreneur/image-data-augmentation-using-keras-imagedatagenerator-1cee60255ea8.
37. paperswithcode.com. (n.d.). Papers with Code - EfficientNet Explained. [online] Available at: https://paperswithcode.com/method/efficientnet.
38. Pratibha Kumari J (2024). Machine Learning with Python: a Comprehensive Guide to Algorithms, Tools, and Best Practices. [online] Linkedin.com. Available at: https://www.linkedin.com/pulse/mastering-machine-learning-python-comprehensive-guide-jha-wlouc/ [Accessed 19 Sep. 2024].
39. Pugliese, R., Regondi, S. and Marini, R. (2021). Machine learning-based approach: Global trends, Research directions, and Regulatory Standpoints. Data Science and Management, [online] 4, pp.19–29. doi:https://doi.org/10.1016/j.dsm.2021.12.002.

40. Ratan, P. (2020). Convolutional Neural Network Made Easy for Data Scientists. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/.
41. Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Computer Science, [online] 2(3), pp.1–21. doi:https://doi.org/10.1007/s42979-021-00592-x.
42. scikit-image.org. (n.d.). scikit-image: Image Processing in Python — scikit-image. [online] Available at: https://scikit-image.org/.
43. Scikit-learn (2019). scikit-learn: Machine Learning in Python. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/.
44. Scipy.org. (2024). Random — SciPy v1.14.1 Manual. [online] Available at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.random.html [Accessed 20 Sep. 2024].
45. seaborn (2012). seaborn: Statistical Data Visualization — Seaborn 0.9.0 Documentation. [online] Pydata.org. Available at: https://seaborn.pydata.org/.
46. Simonyan, K. and Zisserman, A. (2015). Published as a Conference Paper at ICLR 2015 VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. [online] Available at: https://arxiv.org/pdf/1409.1556.
47. Simplilearn.com. (2023). Data Preprocessing in Machine Learning: a Beginner's Guide. [online] Available at: https://www.simplilearn.com/data-preprocessing-in-machine-learning-article.
48. StudySmarter UK. (2019). Object Recognition: Algorithms & Techniques | StudySmarter. [online] Available at: https://www.studysmarter.co.uk/explanations/engineering/robotics-engineering/object-recognition/#:~:text=Object%20recognition%20is%20a%20crucial [Accessed 19 Sep. 2024].
49. Sumbatilinda (2024). Deep Learning (Part 3). CONVOLUTION NEURAL NETWORKS(CNNs). [online] Medium. Available at: https://medium.com/@sumbatilinda/deep-learning-part-3-convolution-neural-networks-cnns-acd07bfeb6a1.
50. Susan Currie Sivek (2024). Mastering Model Complexity: Avoiding Underfitting and Overfitting Pitfalls. [online] Pecan AI. Available at: https://www.pecan.ai/blog/machine-learning-model-underfitting-and-overfitting/.
51. Talebi, H. and Milanfar, P. (2021). Learning to Resize Images for Computer Vision Tasks. [online] arXiv.org. Available at: https://arxiv.org/abs/2103.09950 [Accessed 20 Sep. 2024].
52. Team, K. (n.d.). Keras documentation: LearningRateScheduler. [online] keras.io. Available at: https://keras.io/api/callbacks/learning_rate_scheduler/.
53. Team, K. (2023). Keras documentation: RandomFlip Layer. [online] Keras.io. Available at:

https://keras.io/api/layers/preprocessing_layers/image_augmentation/random_flip/#:~:text=RandomFlip%20class&text=A%20preprocessing%20layer%20which%20randomly.

54. Team, K. (n.d.). Keras documentation: ReduceLROnPlateau. [online] keras.io. Available at: https://keras.io/api/callbacks/reduce_lr_on_plateau/.
55. TensorFlow (2019). Keras | TensorFlow Core | TensorFlow. [online] TensorFlow. Available at: https://www.tensorflow.org/guide/keras.
56. W3Schools (n.d.). Matplotlib Pyplot. [online] www.w3schools.com. Available at: https://www.w3schools.com/python/matplotlib_pyplot.asp.
57. Why (2023). Why CIFAR 10 Vs. CIFAR 100 Is the Most Popular for OOD benchmark? [online] Data Science Stack Exchange. Available at: https://datascience.stackexchange.com/questions/120173/why-cifar-10-vs-cifar-100-is-the-most-popular-for-ood-benchmark [Accessed 19 Sep. 2024].
58. Wikipedia Contributors (2019a). False Positives and False Negatives. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/False_positives_and_false_negatives.
59. Wikipedia Contributors (2019b). Python (programming language). [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Python_(programming_language).
60. Wikipedia. (2021a). Early Stopping. [online] Available at: https://en.wikipedia.org/wiki/Early_stopping#:~:text=In%20machine%20learning%2C%20early%20stopping.
61. Wikipedia. (2021b). False Positive Rate. [online] Available at: https://en.wikipedia.org/wiki/False_positive_rate.
62. www.javatpoint.com. (n.d.). Majority Voting Algorithm in Machine Learning - Javatpoint. [online] Available at: https://www.javatpoint.com/majority-voting-algorithm-in-machine-learning.
63. www.linkedin.com. (n.d.). Machine Learning Vs Traditional Programming: Which Is More Efficient? [online] Available at: https://www.linkedin.com/pulse/machine-learning-vs-traditional-programming-argvf/.
64. www.tutorialsteacher.com. (n.d.). Python Os Module. [online] Available at: https://www.tutorialsteacher.com/python/os-module.
65. www.w3schools.com. (n.d.). Seaborn. [online] Available at: https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp.
66. Zhou, T., Ye, X., Lu, H., Zheng, X., Qiu, S. and Liu, Y. (2022). Dense Convolutional Network and Its Application in Medical Image Analysis. BioMed Research International, 2022, pp.1–22. doi:https://doi.org/10.1155/2022/2384830.