ETHIOPIAN CURRENCY DETECTION AND COUNTERFEIT VERIFICATION USING DEEP LEARNING

A Thesis Presented

by

Mikias Melaku Gugsa

to

The Faculty of Informatics

of

St. Mary's University

In Partial Fulfillment of the Requirements

for the Degree of Master of Science

in

Computer Science

June, 2023

Addis Ababa, Ethiopia

ACCEPTANCE

Ethiopian Currency Detection and Counterfeit Verification Using Deep Learning

By

Mikias Melaku Gugsa

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

____

Internal Examiner

____

External Examiner

Alembante Mulu (PhD)

____

Dean, Faculty of Informatics

June 2023

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Mikias Melaku Gugsa
Full Name of Student

____
Signature
Addis Ababa
Ethiopia

This thesis has been submitted for examination with my approval as advisor.

Alembante Mulu (PhD)
Full Name of Advisor

____
Signature
Addis Ababa
Ethiopia

June 2023

Acknowledgement

Firstly I would like to thanks God for always being there for me and giving me courage and patience in doing this work. It would not have been to this stage without the help of God. Next, I would like to express my deepest gratitude to my thesis advisor Dr. Alembante Mulu for contributing his constructive comments, directions, ideas, and courage during the thesis. I would like to give my special thanks to my Wife for her encouragement and support all through my studies and throughout my life. Finally, I would like to extend my sincere thanks to all of my families, friends, and Dr. Dereje Tekilu Aseffa and Commander Demis, for sharing me Ethiopian genuine and counterfeit banknote dataset.

Table of Contents

List of Acronyms

ANN	Artificial Neural network
ATM	Automatic Teller Machine
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DRE	Dynamic Range Adjustment
F10	Fake 10 Ethiopian banknote
F100	Fake 100 Ethiopian banknote
F200	Fake 200 Ethiopian banknote
F5	Fake 5 Ethiopian banknote
F50	Fake 50 Ethiopian banknote
FN	False Negative
FP	False Positive
G10	Genuine 10 Ethiopian banknote
G100	Genuine 100 Ethiopian banknote
G200	Genuine 200 Ethiopian banknote
G5	Genuine 5 Ethiopian banknote
G50	Genuine 50 Ethiopian banknote
GA	Genetic Algorithm
GPU	Graphical Processing Unit
RELU	Rectified Linear Unit
ResNet	Residual Neural Network
RGB	Red, Green, and Blue
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

List of Figures

List of Tables

Abstract

The technology of currency identification plays a vital role in automated self-service equipment such as ATMs, vending machines, and smart card charging machines. These devices require accurate banknote recognition, counterfeit detection, serial number recognition, and fitness classification. However, existing banknote detectors are often tailored to specific countries and cannot be easily reprogrammed for currency identification. Moreover, banknote recognition algorithms based on deep learning suffer from small training datasets and lower accuracy.

In this study, we address these challenges by focusing on Ethiopian banknotes. We collected a diverse dataset of Ethiopian real and counterfeit banknotes, including varying ages and conditions. The dataset size and quality significantly impact the performance of the recognition system. To extract effective features from the banknotes, we employed convolutional neural networks (CNNs) using popular architectures such as InceptionV3, VGG16, MobileNet, and ResNet50.

We conducted experiments with different optimization approaches, including Adam and Stochastic Gradient Descent (SGD). These optimization approaches influence the training process and model performance. Additionally, we compared the accuracy of the models to determine the most effective solution for Ethiopian banknote identification.

Our evaluation metrics included accuracy, which measures the overall correctness of the banknote recognition system. Among the models tested, MobileNet trained with SGD optimization and a batch size of 32 achieved the highest training accuracy of 99.6% and overall accuracy 97%. This outperformed the other deep learning models considered in this study. The MobileNet model with SGD optimization is implemented in both a web-based application and Android applications designed specifically for Android mobile devices.

This research contributes to the development of a reliable system for Ethiopian banknote identification. By leveraging deep learning techniques and optimization approaches, we address the limitations of existing systems, such as small datasets and lower recognition accuracy. The findings demonstrate the potential of using MobileNet with SGD optimization as an effective solution for banknote recognition in Ethiopia, paving the way for improved currency identification in automated self-service equipment.

Keywords: *Ethiopian Currency, Currency Recognition, Counterfeit Detection.*

CHAPTER ONE

1. INTRODUCTION

1.1 Background of the Study

During Ethiopia's modern history, the Bank of Abyssinia introduced the country's first banknotes in 1915. These early banknotes had denominations of 5, 10, 100, and 500 talari and remained unchanged until 1945.

In 1945, Emperor Haile Selassie issued a decree mandating the use of new Ethiopian banknotes featuring his likeness. These new banknotes were introduced in denominations of 1, 5, 10, 50, 100, and 500 birrs. In 1976, the government under DERG made significant changes to the banknote designs, including the removal of the emperor's features. Subsequently, in November 1997, the government controlled by EPRDF introduced minor modifications to the DERG currency, such as changes in features and colors. Notably, the higher denominations of 50 and 100 birrs received enhanced security features during this time.

In the year 2020, Ethiopia's government unveiled a fresh set of currency notes that incorporated advanced security measures and unique characteristics. These new banknotes replaced the previously circulating 10, 50, and 100 birr notes, while also introducing a new denomination of 200 birrs. The design and features of the new currency notes are entirely distinct from those used 46 years ago. The newly introduced banknotes were carefully crafted to showcase the country's natural and historical aspects while also reflecting its aspirations for future development, including industrialization and modern agriculture. These features were incorporated to create a currency that not only meets the requirements of a reliable medium of exchange but also represents Ethiopia's rich heritage and ambitious vision.

Currently, in Ethiopia, paper money remains a primary means of exchanging goods and services. However, the persistent challenge lies in detecting counterfeit banknotes that closely resemble genuine ones, making it difficult for non-experts to identify them. While there are counterfeit detection machines available, they are often expensive, leading to financial and government entities shouldering the responsibility of identifying and confiscating counterfeits, with limited involvement from the community.

Traditionally, people relied on their naked eyes to determine the authenticity of money. However, human visual perception has limitations, and distinguishing genuine banknotes from counterfeits becomes challenging without leveraging new technologies. Although UV recognition technology is already in use, it is insufficient to combat increasingly sophisticated counterfeiting techniques. Fortunately, with advancements in image recognition, researchers have explored specific identification methods by analyzing currency color and specific data through image analysis and enhancement. By employing data augmentation techniques, such as color analysis, researchers expand the dataset used for currency recognition methods. This approach allows for more accurate identification of counterfeit currency, overcoming the limitations of human visual perception and UV recognition technology.

Initially, a large dataset is required to initiate the process. Through careful analysis of this training dataset, the accuracy of currency recognition can be steadily enhanced, aligning with our anticipated experimental outcomes. In this recognition process, the utilization of Convolutional Neural Network (CNN) [1] proves to be pivotal, as it significantly improves the overall training accuracy. To accomplish this, we will employ CNN as a feature extractor within the framework of the Single Shot Multi-Box Detector (SSD) model [2].

During the currency recognition process, it is crucial to assess the adequacy of the dataset size. Since our data collection involved scanning multiple images at a time, there is a possibility of distortion or blurring. Therefore, it becomes necessary to edit the images and enhance their clarity, as this significantly contributes to improved accuracy after training.

In the realm of deep learning, the issue of overfitting must be addressed. It is important to prevent the training process from becoming excessively complex, which can lead to increased training difficulty and duration. Implementing techniques like dropout can help mitigate overfitting and ensure optimal training outcomes.

1.2 Motivation

The motivation for this research stems from various factors:

Absence of Automatic Teller Machines (ATMs): In Ethiopia, the lack of ATMs capable of accepting cash deposits into personal or other customers' accounts is evident. Such ATMs require robust mechanisms to differentiate between counterfeit and genuine banknotes. This research aims to address this need by developing reliable currency recognition systems that can be integrated into ATMs, enabling secure and efficient cash transactions.

Advancement of Automated Systems: This research also seeks to contribute to the advancement of automated systems in Ethiopia, including vending machines, ticket counters for buses and railways, note counters in the banking system, shopping malls, and currency exchange services. These automated systems are currently not widely available in the country. By developing accurate currency recognition techniques, this research aims to facilitate the implementation of such automated systems, enhancing convenience and efficiency in various sectors. For example, in the banking sector, the ability to quickly and accurately identify banknotes would streamline the process of cash handling, reducing human error and increasing operational efficiency. Similarly, in shopping malls and currency exchange services, automated machines equipped with reliable currency recognition capabilities would ensure smooth transactions and prevent the circulation of counterfeit money.

Limited Research in Ethiopia: While currency recognition research is being conducted worldwide, there is a lack of studies specifically focused on Ethiopian banknotes. This research aims to fill this gap by conducting a comprehensive investigation into Ethiopian currency recognition. By examining the unique features and characteristics of Ethiopian banknotes, this research will contribute to the knowledge base in the field of currency recognition, specifically tailored to the Ethiopian context.

The motivations behind this research lie in addressing the absence of currency recognition systems in ATMs, facilitating the implementation of automated systems in various sectors, and contributing to the limited body of research on Ethiopian banknotes. By achieving these goals, this research aims to enhance financial security, improve operational efficiency, and promote technological advancements in Ethiopia.

1.3 Statement of the problem

In Ethiopia, bankers currently employ a semi-automated technology called the UV-light detector and counterfeit detector pen to identify and validate fake banknotes. By flashing ultraviolet light on a banknote, the UV light technology makes previously unseen security elements apparent and easily identifiable. The counterfeit detecting pen, on the other hand, works by writing on the banknote and watching to see whether the pen reacts with it. If it does, the banknote is classified as false. Nonetheless, because there are so many banknotes in circulation, it is impossible to screen for counterfeits every time. Furthermore, counterfeit banknote detectors are extremely expensive, and finding counterfeit banknotes is difficult. Additionally, both methods are semi-automated and hardware-based, requiring human participation to validate the banknotes' authenticity. Furthermore, for large-scale and secure financial transactions, the employment of an automated machine has become necessary. These circumstances need the development of a full-fledged software-based automated currency transaction system that allows ATMs to execute cash-in and cash-out services.

The approach used to extract the discriminative feature connected to the characteristic feature accessible in the banknotes and the characteristic feature that makes up the banknotes are intimately connected in the automatic banknote identification system. For example, some banknotes, such as the US dollar, use photo chromatic materials (color-changing materials that change color when viewed from different angles) to identify them. Recognizing a banknote with a photo chromatic security feature necessitates a technique that considers the chromatic nature of the paper note. Furthermore, other nations' Indian banknotes, such as the US dollar and Chinese banknotes, have unique serial number that is known and used for verification and recognition [3].

The Ethiopian National Bank adopted a difficult pattern and security features to differentiate the birr note as shown Figure 1.1, and the design necessitates a demanding feature descriptor that can grasp the birr note image's strong pattern. According to [4], presentation of the feature direction, to attain a high identification rate, There is a need for an enhanced feature extraction and classification method in the recognition of Ethiopian paper currency. Due to variations in the quality of paper used, it is difficult to identify an appropriate feature extraction and classification strategy that takes these differences into account.

The objective of this study is to explore and examine alternative feature extraction techniques to build a model for Ethiopian banknote recognition and detection.

As a matter of fact, there is no common technique that works efficiently across countries [5] . Countries throughout the world used a variety of security measures to create their own banknotes, including visible and invisible security markings, watermarks, images that reflect their historical location and history, and many more symbols and iconography. And the capacity to recognize banknotes is more likely linked to the security measures used in their production and the methodologies used to extract characteristics [6]. Because of these differences in features, none of the approaches can ensure that they will apply across all currencies.

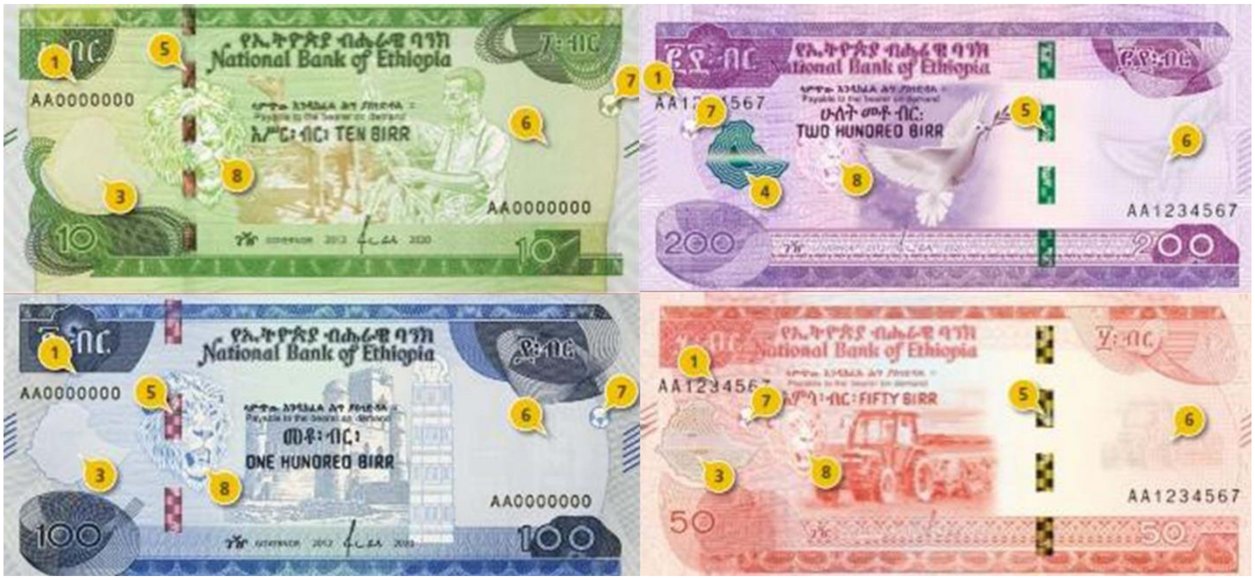


Figure 1- 1. Ethiopian Paper currency front page with security feature indicator

1.4 Research question

- Which approach of Convolutional Neural Network is best for recognizing Ethiopian banknotes?
- What banknote security elements can be utilized to detect genuine and counterfeit Ethiopian banknotes?
- To what level does the model effectively classify the picture of a paper currency note into the right class and fake currency notes?

1.5 Objective of the study

1.5.1 General Objective

The general objective of this study is to achieve an Ethiopian banknote detection and counterfeit verification using deep leaning that can accurately identify Ethiopian bank note denominations as well as detect counterfeit currency and develop web based and android application (software) for user interface.

1.5.2 Specific Objectives

To achieve the research objective, the following specific objectives are addressed in this study.

- To review literatures on currency recognition systems, major processes of currency recognition systems and different countries' currency recognition approaches.
- To collect various genuine and counterfeit Ethiopian banknotes.
- Apply various optimization techniques for CNN architectures to identify the best optimization techniques.
- To develop Web based and android app for user interface.
- To assess the effectiveness of the model using suitable evaluation metrics.

1.6 Scope of the study

The objective of this research is to develop a model capable of analyzing photographs of Ethiopian banknotes, specifically 5-birr, 10-birr, 50-birr, 100-birr, and 200-birr denominations. The model selection process involves identifying the most suitable neural network architecture. Subsequently, the model will be prepared and datasets will be created, including a training dataset, validation dataset, and test dataset. The model will be trained using the training dataset, with parameter and hyper parameter tuning conducted using the development dataset to identify optimal values. The final model's performance will be evaluated using appropriate metrics on the test dataset. Additionally, the model will categorize the banknote images into different classifications, such as counterfeit or genuine. Notably, this study does not cover the recognition of Ethiopian coin currency and is not intended for individuals with visual impairments.

1.7 Methodology

This study follows an experimental research approach with the objective of designing and developing a model for Ethiopian banknote denomination recognition and counterfeit currency detection. The methodology involves the following steps:

1.7.1 Conducting a literature review

The first step is to review existing techniques and methodologies for currency detection and counterfeit verification using deep learning. This helps to understand the state-of-the-art approaches and identify gaps that the research aims to address.

1.7.2 Dataset collection

A diverse dataset of Ethiopian currency images is collected, encompassing both genuine and counterfeit notes of various denominations and conditions. This dataset ensures the representation of real-world scenarios and challenges encountered in currency recognition.

1.7.3 Preprocessing of currency images

The collected currency images undergo preprocessing to enhance their quality and ensure consistency in terms of size and format. This step improves the input data for the deep learning model and helps in achieving accurate results.

1.7.3 Selection of deep learning model architecture

A suitable deep learning model architecture, such as a convolutional neural network (CNN), is chosen for currency detection and counterfeit verification. The selected model should be capable of learning relevant features and patterns from the currency images.

1.7.4 Training the model

The chosen deep learning model is trained using the prepared dataset. During training, the model's performance is optimized through iterations and fine-tuning of hyper parameters. This process aims to improve the model's accuracy and ability to differentiate between genuine and counterfeit currencies.

1.7.5 Evaluation of model performance

The trained model is evaluated using a separate test dataset. The performance is measured using appropriate evaluation metrics such as accuracy, precision, recall, and F1 score. This step assesses how well the model can accurately detect Ethiopian currencies and verify their authenticity.

1.7.6 Experimental validation

Experiments are conducted to validate the effectiveness of the proposed model in detecting Ethiopian currencies and verifying their authenticity. The experiments involve testing the model on real-world scenarios and evaluating its performance in different situations.

1.7.7 Analysis and comparison of results

The obtained results are analyzed and compared with existing approaches in the field of currency recognition and counterfeit detection. This analysis highlights the strengths and limitations of the proposed methodology and provides insights into its performance compared to other techniques.

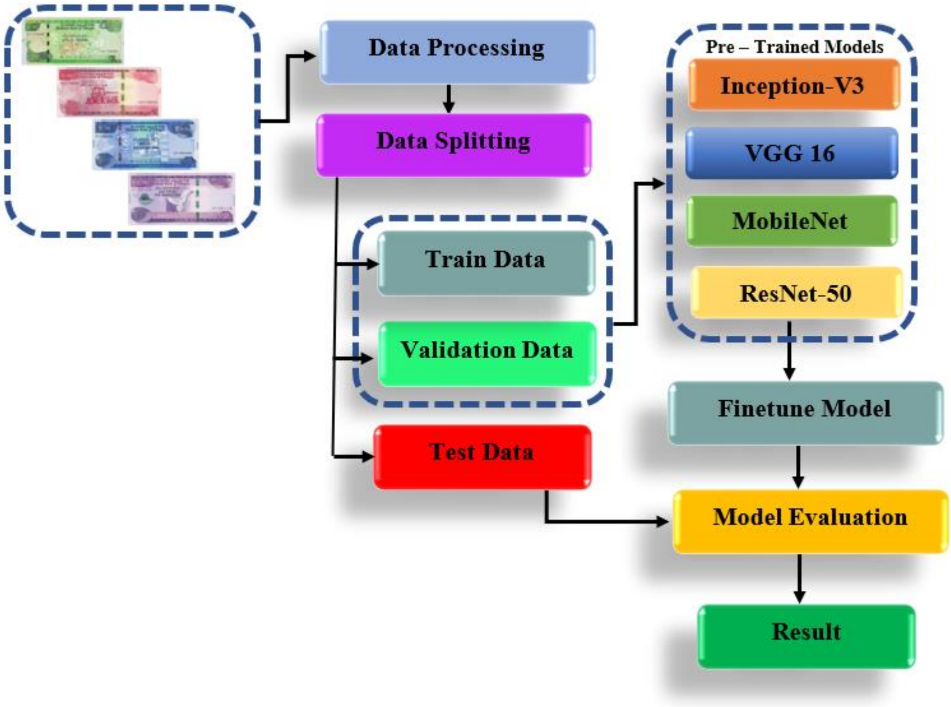


Figure 1- 2. Workflow of the proposed counterfeit detection method

1.7.8 Conclusion and future research

The findings of the study are summarized in the conclusion section, discussing the achievements, limitations, and implications of the developed model. Suggestions for future research directions are also provided, aiming to further enhance currency recognition and counterfeit detection methods.

1.8 Significance of the study

The aim of this thesis is to design a model for Ethiopian banknote classification and counterfeit verification system for automating the money transaction system. The designed system is an input for self-serving devices like ATMs, self-servicing Kiosks, Food, and Beverage dispensers by making them intelligently recognize banknotes. Since the designed system identifies fake currencies from genuine ones, which will be a great solution for the banking industry and society to identify counterfeit. This will also be helpful for police and intelligent agents to investigate the criminals who are engaging in money counterfeiting activities. We anticipate that this research will yield various benefits to the target populations within the identified domain, People who are engaging in Ethiopia paper currency circulation and institutions who have self-serving devices like bankers will be benefited. Moreover, the research will be referred by future researchers in the domain area.

1.9 Thesis Organizations

This thesis is structured into five main chapters, namely the introduction, literature review, system design and modeling, result & discussion, and conclusion & future works. The introduction chapter provides a concise background of the study, outlining the motivation behind the research, stating the problem to be addressed, specifying the objectives, describing the methodology employed, defining the scope of the study, and highlighting its significance. The literature review chapter presents an extensive review of relevant literature and previous research works that are related to the problem being investigated. It aims to provide a comprehensive understanding of the existing knowledge and research gaps in the field. The system design and modeling chapter delves into the detailed design of the model for the proposed system, specifically tailored to address the identified problem. This chapter outlines the architecture, components, and functionalities of the proposed system. In the Experiment and discussion chapter, various experiments are conducted using selected fine-tune model, and the performance of the developed models is thoroughly analyzed and discussed. This chapter presents the results obtained from the experiments, providing insightful discussions and interpretations. Finally, the conclusion & future work chapter summarizes the results and findings of the research. It consolidates the main outcomes and draws meaningful conclusions. Additionally, this chapter identifies potential areas for future research and suggests directions for further investigations in the field.

CHAPTER TWO

2. LITERATURE REVIEW

2.1 Overview

The rise in circulation of counterfeit money has raised concerns about the effectiveness of the money identification system. This chapter aims to address these concerns by evaluating the current practice of recognizing Ethiopian local banknotes. It begins by discussing the existing methods employed in the recognition of banknotes in Ethiopia. Furthermore, a comprehensive review of relevant literature and studies conducted by other researchers on banknote identification systems is presented. The chapter then delves into an in-depth explanation of deep learning, neural networks (NN), and convolutional neural networks (CNN). It explores their definitions and concepts. Additionally, the researcher provides a detailed description of banknote classification and demonstrates how it can be achieved using a Convolutional Neural Network (CNN).

2.1 Theoretical Literature Review

2.1.1 Existing technique for banknote recognition in Ethiopia

To address the counterfeiting issue, different strategies including first-line inspection methods and second-line inspection methods are being applied in various countries. People in most countries of the world use the first-line inspection approach, which involves just inspecting some of the security measures that make up the banknote. For example, in Ethiopia, both bank officials and regular customers used a first-line inspection technique to identify local domestic Ethiopia banknotes by looking at some of the most common security features like the security thread and golden stripes, the identifying symbol, as well as the numerical denomination, is written in Arabic and geez.

The other inspection procedures, which are classified as automated and semi-automated, recognize banknotes. In most countries, individuals utilize both automated and manual inspection techniques to recognize banknotes in their daily activities. The following are some semi-automatic inspection techniques: Counterfeit Detection pen, ultraviolet light detector, and counterfeit detection scanner [7]. Currently, the commercial Bank of Ethiopia, non-governmental banks, and financial institutions in Ethiopia use counterfeit detecting pens and ultraviolet light detectors to distinguish foreign banknotes.

2.1.2 Counterfeit Detection Pen

The Money Tester Pen, also known as the Patented Smart Money Counterfeit Detector Pen, has brought about a significant transformation in the field of counterfeit detection. It is an instrument utilized to assess the authenticity of banknotes, distinguishing between genuine ones and counterfeit ones. A tincture of iodine solution is included in the pen, which interacts with the starch in the wood-based paper to produce a black stain. When applied to the fiber-based paper used in real banknotes, there is no color. The iodine in the pen interacts with starch, brightening the white paper. There will be no starch and the pen will not leave a mark if the bill is genuine and the paper is fiber-based [7]. It is mostly used in Ethiopia to recognize the originality of both international and domestic banknotes by the Commercial Bank of Ethiopia, non-governmental banks, and financial organizations. However, because it requires human participation to identify banknotes, it is a semi-automated and hardware-based technology.

2.1.3 Ultraviolet counterfeit detection scanner (UV)

Most currencies include ultraviolet security features, which the UV detector detects. Counterfeit cash is automatically discovered by simply inserting the note in the detector, without the need for an employee to extensively study the note. When exposed to UV light, the UV-printed pictures should shine, indicating that the banknote is genuine. Similar currency detecting tools, such as a magnetic counterfeit detector, use magnetic detection to spot magnetic ink and metal threads inserted in selected locations of banknotes to determine their validity. Watermark counterfeit detectors, which utilize light to examine watermarks embedded in genuine banknotes, are also available. Counterfeits will be immediately rejected. The UV counterfeit detection scanners are the most affordable and widely accessible of these. There are security measures on Ethiopia banknotes that are created utilizing UV-printed pictures that can only be seen when exposed to UV light. But it is also affected by noise. Furthermore, the watermark security feature is used, which may be identified with watermark counterfeit detectors. The recognition of the banknote, however, was not totally automatic and required human involvement.

2.3 Image processing for paper currency recognition

Image processing refers to the method of converting an image into a digital format and conducting various operations to enhance the image quality or extract significant information from it. It encompasses tasks such as image preparation, analysis, and understanding. Banknote recognition systems are one application that benefits from image processing techniques. In any currency identification system, there are five key steps involved: image capture, pre-processing, segmentation, feature extraction, and classification. These steps collectively contribute to the accurate identification and analysis of banknotes [24]. The development of an image classification and object recognition system involves various phases in image processing. It is not mandatory to understand all the processes for picture recognition, as the stages are tailored to address specific problems. The phases of image processing include image capture, preprocessing, edge detection, segmentation, morphological analysis, feature extraction, and feature selection. According to [4], the process of banknote recognition systems involves several essential steps, including image acquisition, pre-processing, Image segmentation, feature extraction, classification, and recognition.

2.3.1 Image acquisition

The process of obtaining an image from a source, typically a hardware-based device, can be broadly defined. Picture capture is invariably the first step in the workflow sequence, as image processing cannot proceed without an initial image. It involves capturing a picture from a real-life scene, ensuring that the captured image preserves all the distinctive characteristics of the banknotes.

2.3.2 Pre-processing

The quality of an item's image can degrade due to the presence of noise on the object's surface, which may result from factors such as aging, scratches, scaling, and other types of distortion. These factors significantly affect the performance of recognition systems. To mitigate the impact of noise on an image, it is crucial to apply substantial preprocessing techniques and employ suitable noise filtering methods. In addition, there is no single noise filtering algorithm that can effectively address all types of noise present in paper note images [25]. Additional common preprocessing operations involve normalizing the size and brightness of images, as well as correcting skewed angles and performing channel transformations. Skewness specifically pertains to the inclination present in digitally scanned images of paper money for optical character recognition (OCR). Skew angle detection is a crucial aspect of OCR systems and document analysis. When digitizing banknote images with high-speed scanners, significant tilt can occur due to manual or automatic feeding. The skew angle represents the angle of inclination that is introduced during the scanning process. The presence of a skew angle in the recognition of banknote serial numbers can be a common occurrence due to misaligned paper notes. This skew angle can negatively affect the entire identification process and pose challenges in segmenting the banknote serial number accurately. Therefore, it is crucial to detect and correct the skew angle before proceeding to the next phase of serial number identification using OCR. Once the skew angle has been determined, the banknote is simply rotated in the opposite direction by the calculated angle to rectify the skewness [26].

2.3.3 Image Segmentation

Image segmentation refers to the process of dividing a digital image into multiple segments or sets of pixels, also known as super pixels. The primary objective of segmentation is to enhance the understanding and analysis of an image by simplifying and/or modifying its representation. Image segmentation plays a crucial role in identifying objects and boundaries such as lines and curves within images. Various segmentation methods utilize two fundamental properties of pixels concerning their local neighborhoods: discontinuity and similarity/homogeneity. Edge-based or boundary-based segmentation techniques focus on the discontinuous properties of pixels, while region-based segmentation approaches rely on the concepts of similarity or homogeneity among pixels. According to [27], in many cases, none of the segmentation techniques consistently yield reliable segmentation results, particularly when applied to low-resolution photos that exhibit poor texture quality.

2.3.4 Feature Extraction

Feature extraction is a dimensional reduction technique that plays a significant role in image analysis. Since image data is complex and characterized by multiple dimensions, it becomes essential to extract informative features from the images to facilitate tasks such as object detection and segmentation [28]. Feature extraction is a challenging task in banknote image processing tasks. The objective in this context is to analyze and identify the distinct and discriminative characteristics of each banknote denomination, even under challenging conditions such as old or worn-out notes, varying illumination, and diverse backgrounds. According to [28], the feature extraction approach aims to extract relevant information from a large set of visual data while preserving as much useful information as possible. In the case of banknotes, various methods are employed to extract features by examining individual components such as color, texture, size, and shape. These methods enable the extraction of meaningful features from banknotes for further analysis and processing [4].

2.4 Deep Learning

Deep learning, particularly the use of convolutional neural networks (CNNs), has emerged as a powerful technique for banknote detection and classification in image processing applications.
CNNs are a type of multilayer deep learning neural network that excels at feature extraction from images [11]. They consist of multiple layers that act as an automated feature extractor and a trainable classifier. By leveraging hierarchical representations, CNNs can extract relevant topological attributes/features from banknote images. In a CNN, the initial layers perform local feature extraction by analyzing low-level image components such as edges, textures, and shapes. As the network progresses deeper, higher-level features such as specific patterns and colors associated with different banknote denominations are learned. Ultimately, the last layer of the CNN classifies the pattern extracted from the input image. The strength of CNNs lies in their ability to manipulate pixels at various levels, allowing them to extract features from complex and high-dimensional data. The convolution and pooling layers in a CNN play crucial roles in capturing and learning meaningful patterns from banknote images. These layers apply convolutional operations to detect important features and then down sample the feature maps using pooling operations. CNNs also incorporate other layers, such as activation and dropout layers, to enhance their performance. Activation functions introduce non-linearity, enabling the network to learn complex relationships between input features and output classes. Dropout layers help prevent overfitting by randomly disabling some neurons during training, thereby increasing the model's generalization ability.

By training CNNs on large datasets containing annotated banknote images, the network can learn to differentiate between genuine and counterfeit banknotes. The training process involves optimizing the network's parameters and weights to minimize classification errors, typically using optimization algorithms like stochastic gradient descent. The trained CNN is then evaluated on a separate test dataset to assess its performance in accurately classifying banknotes.

It is worth noting that CNNs are just one example of deep learning approaches for banknote detection. Other models, such as recurrent neural networks (RNNs) or hybrid architectures, can also be explored to capture temporal dependencies in sequences of banknote images or leverage multiple modalities of information for improved detection accuracy.

2.5 Feature extraction techniques

Color, shape, size, and texture are crucial attributes of paper money and are frequently discussed in the context of banknote recognition research. This section details the process of extracting color, shape, and texture characteristics from banknotes [4].

2.5.1 Colour feature extraction

When observing an image, color is the most prominent and significant aspect that humans perceive. Due to the human visual system's higher sensitivity to color information compared to grayscale levels, color becomes the primary consideration for feature extraction [29]. When working with color information in an image, there are two aspects to consider: the color distribution and the spatial positioning of the colors. The choice of color space significantly impacts the performance of the resulting color feature. Among the various color spaces utilized in computer graphics, photo categorization, and identification applications, the HSV color space is widely adopted. The HSV color model, also known as HSB (Hue, Saturation, Brightness), defines a color space based on three key components: Hue, Saturation, and Value. In this color space, Hue is utilized for color discrimination, Saturation represents the degree of white light mixed with a pure color, and Value denotes the perceived light intensity, ranging from zero for black to one for white [30] [4]. The hue component of the HSV color space remains unaffected by variations in illumination and camera direction. The HSV color space offers the advantage of being more aligned with the human perception and understanding of colors. It enables the separation of chromatic (color-related) and achromatic (non-color-related) components, enhancing the ability to differentiate between different colors in an image [31]. The Hue signal in the HSV color space corresponds to human color perception and is less susceptible to noise interference. On the other hand, the Saturation and Value components of the HSV color space are more prone to noise contamination [31]. Relying solely on the color feature of paper notes is insufficient to achieve accurate banknote recognition due to its sensitivity to various factors such as illumination variations, changes in appearance over time, and improper handling of the notes.

Many researchers utilize color histograms to extract the color feature of banknote images, as they provide valuable insights into the color distribution within the image. Color histograms serve as discriminative features that enable differentiation between different types of images. However, when it comes to image classification, color histogram methods face certain challenges. One such challenge is the accuracy in describing the appropriate color resolution. When using a low color resolution with only a few bins, it may not provide enough color depth to effectively differentiate between various types of images. On the other hand, increasing the number of bins can lead to greater computation complexity for similarity calculations and require more memory storage for the database [32].

2.5.2 Shape feature extraction

In addition to color and texture data, the shape of objects is often utilized in image categorization and recognition. Various researchers explore different methods for representing the shape of an image [33]. It can be categorized into two groups: contour-based methods, which calculate shape features solely from the boundaries of objects, and region-based methods, which extract shape features from the entire object region. The shape feature descriptors involve various measurements such as area, circularity, eccentricity, major axis orientation, and bending energy. Global boundary descriptors include different types of signatures, Fourier descriptors, and wavelet descriptors. Regions can be described using simple geometric parameters like area or compactness. Grid-based and moment invariants methods are among the commonly used and popular region descriptors. However, in the context of Ethiopian banknotes, the shape feature may not be practical due to its sensitivity to noise and paper quality. Shape features are expected to be invariant to scaling, rotation, and translation of objects in image recognition. However, the development of shape features is less advanced compared to color and texture features, mainly due to the inherent complexity involved in representing shapes [34].

2.5.3 Texture feature descriptor

Texture refers to the variations in surface intensity and is a crucial measure in assessing properties like smoothness, coarseness, and regularity. It is commonly used as a region descriptor in computer vision and image analysis. Various techniques, such as ORB, SIFT, SURF, BRISK, BRIEF, HARRIS, FAST, and MSER, are available for handling texture in images. However, when it comes to banknotes that undergo frequent circulation, develop wear and tear, and possess limited texture, the SURF local feature extraction technique may not be the most suitable approach. Additionally, SURF feature descriptors may not be ideal for detecting features in complex patterns within images.

2.5.4 CNN (Convolutional neural network) feature extractor

Deep Neural Networks (DNNs), also referred to as Multilayer Perceptions or Feedforward Neural Networks, are artificial neural networks with multiple hidden layers. They are capable of learning hierarchical features in an automatic and adaptive manner through the backpropagation algorithm. DNNs consist of different types of layers in the hidden layer, including convolutional layers, pooling layers, and fully connected layers, which serve as functional building blocks. These networks possess several advantages, such as a simple structure, a smaller number of training parameters, and the ability to adapt to different tasks. The weight-sharing structure of DNNs resembles that of biological neural networks. The performance of convolutional neural networks (a type of DNN) is directly influenced by their architectural design [35].

2.6 The Architecture of the CNN model

When designing the architecture of a CNN network, it is necessary to carefully set the parameters for the convolutional layer, pooling layer, and fully connected layers. These parameters play a crucial role in the network's performance. However, determining the optimal values for these hyper parameters often involves empirical analysis, where different combinations are tested until the network starts training effectively [36]. And, it also has an influence on the overall performance of the CNN model [37]. The choice of feature detector size in the convolutional layer is closely linked to the region of interest (ROI) that we aim to detect. If we intend to extract small security features from banknotes, using a larger filter size may result in the loss of these features. It is crucial to carefully consider the filter size to ensure effective detection of the constituent components. Therefore, the selection of parameters during the design of the convolutional layer significantly affects the overall performance of the CNN in recognition tasks [36].

2.7 Transfer Learning

Transfer learning is an approach in machine learning that tackles the challenge of training networks when data is limited. It involves utilizing pre-trained models that have been trained on large datasets, such as ImageNet, and applying them to similar tasks. This technique has demonstrated success in enhancing the performance of deep neural networks and addressing complex problems in computer vision. By reusing a model trained for one task on another related task, it reduces the need for extensive re-training or fine-tuning. For example, in the case of detecting counterfeit banknotes with limited data, a convolutional neural network model pre-trained on the ImageNet dataset can serve as a starting point for task-specific learning. Deep learning models typically require a substantial amount of labeled data to learn specific patterns, but the scarcity of labeled data poses a challenge, particularly in counterfeit banknote imaging. Transfer learning allows leveraging pre-trained models that have learned from larger datasets when training data for a specific task is limited.

In conclusion, automated money transaction machines heavily rely on banknote recognition and classification systems. The existing research has made noteworthy contributions to this field, despite some limitations and gaps. These limitations include small datasets, specialized system knowledge requirements, and high implementation costs. Additionally, some research involves longer processing times due to complex computational algorithms and internal architecture layers. Moving forward, future research should address issues such as folding, lighting conditions, processing time, paper currency quality, and worn currency notes. Furthermore, while different neural network architectures have been explored, optimization techniques have not been adequately considered. The efficient implementation requirements, such as the type and specifications of computer systems needed, have also been overlooked. Although most researchers evaluate their models on high-speed computers, the cost and complexity of such systems limit their practical usage. Therefore, it is essential to focus not only on software programs, algorithms, and architectures but also on implementing them on lightweight embedded systems to achieve high processing speed with minimal validation time. Consequently, there is a need to develop accurate currency recognition models using CNN classifiers with efficient feature extraction and optimization techniques. As part of this research project, a CNN architecture was employed to design an efficient Ethiopian banknote counterfeit verification system.

2.8 Related Works

2.8.1 Other country banknotes recognition

In an extensive investigation, researchers explored currency recognition systems using advanced techniques. They focused on extracting unique color characteristics of Indian currencies in the HSV color space [13]. Another study extracted features of Euro banknotes from image segments [14]. For Korean banknotes, a novel technique using the visible light spectrum was used, and the Sobel filter was applied to extract wavelet domain features. The K-nearest neighbor method was employed for recognition and classification of Korean won currency notes [15]. A Canadian dollar scanner was developed with pre-recorded voice output to inform users about the denomination value [CNIB, 1996]. A classification system for Sri Lankan rupees was created using a linear transformation function, edge detection algorithms, and backpropagation neural networks [16]. A bill recognition technique achieved 100% accuracy for 1000 and 500 Bangladesh banknotes using the Histogram of Gradient approach and SVM classification [17]. Proposed systems for Slovak EURO currency recognition faced limitations in accuracy due to factors like image positioning, brightness, and quality [18]. A Nigerian currency classification system used SVM learning with genetic algorithm optimization but lacked efficient features and evaluation on other banknotes [19]. A recognition system for Jordanian paper currency processed colored and grayscale images but faced challenges with illumination conditions and exhibited slow performance on cell phones [20]. In machine learning, the CNN with RMSProp algorithm and Albeahdili's PSOSGD methodology showed promising results, and alternative optimizers like Adagrad, Proximal Adagrad, Adam, and RMSProp were used to enhance classification performance [21] [22] [23].

2.8.2 Ethiopian banknotes recognition

Research on Ethiopian banknotes aimed to automate the recognition and detection, with one study using feature extraction and MATLAB to compare currency characteristics [4]. Convolutional Neural Networks with the MobileNetV3 model were utilized to detect Ethiopian banknotes, particularly focusing on surface security elements for authentication [8] . Various methodologies, including SIFT, GLCM, color momentum, and CNN, were employed to enhance feature extraction from images [9]. Another researcher used a feed-forward artificial neural network and pre-trained models like TensorFlow object detection API, Faster R-CNN, Inception, and SSD MobileNet to recognize Ethiopian currency, with a focus on individuals with visual impairments. Evaluation included folded and damaged banknotes, but the processing time for predictions was not addressed [10]. Counterfeit currency identification was explored using image processing techniques and SVM to differentiate Ethiopian banknotes from those of other countries [11]. However, specific research on developing a counterfeit verification system for the new Ethiopian banknotes was lacking [12].

CHAPTER THREE

3. SYSTEM DESIGN AND MODELING

3.1 Introduction

This section covers the proposed model architecture, Model design, dataset preparation, and organization of those datasets. The model has different components to perform the task and it uses different datasets for training and testing of the model. The model has three main phases: training phase, validation, and testing phases. Finally, the model has been evaluated using different evaluation metrics whether it performs well or not. We have used performance metrics of accuracy, precision, F1-score, and confusion matrix for evaluation of the performance of the model.

3.2 System Architecture

The system architecture for Ethiopian currency detection and counterfeit verification using a deep learning consists of five main components: Data set Collection and Preprocessing, Pre-trained Model Selection, Fine-tuning the Model, Model Evaluation, and The Architecture of the Proposed CNN Model.

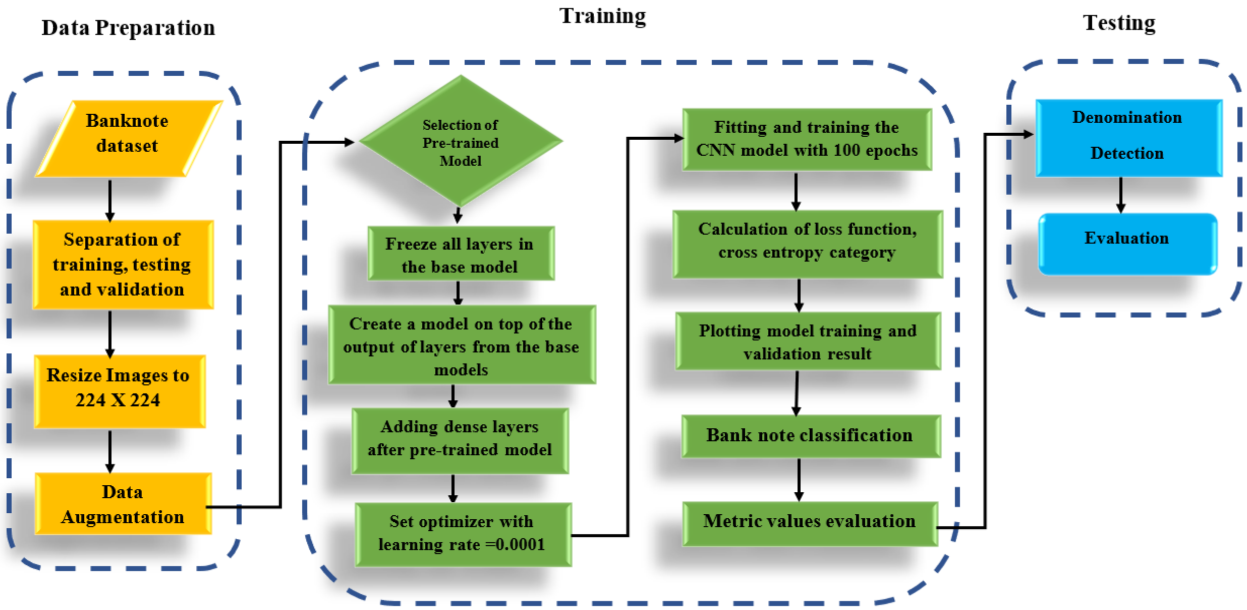

Data Preparation
Training
Testing
Banknote dataset
Separation of training, testing and validation
Resize Images to 224 X 224
Data Augmentation
Selection of Pre-trained Model
Freeze all layers in the base model
Create a model on top of the output of layers from the base models
Adding dense layers after pre-trained model
Set optimizer with learning rate =0.0001
Fitting and training the CNN model with 100 epochs
Calculation of loss function, cross entropy category
Plotting model training and validation result
Bank note classification
Metric values evaluation
Denomination Detection
Evaluation

Figure 3- 1. Proposed model fine-tuning architecture

3.3 Dataset Collection and Preprocessing

The input data collection and processing component includes data collection, data annotation, data cleaning and processing, data splitting and data balancing.

Data Collection: Collecting a diverse and representative dataset of banknote images is the first step in training a model for counterfeit detection. The dataset is contained both genuine and counterfeit banknotes with various denominations, conditions. The banknote images are captured using high-quality mobile cameras and scanners, and the dataset are including images of both the front and back of the banknotes.

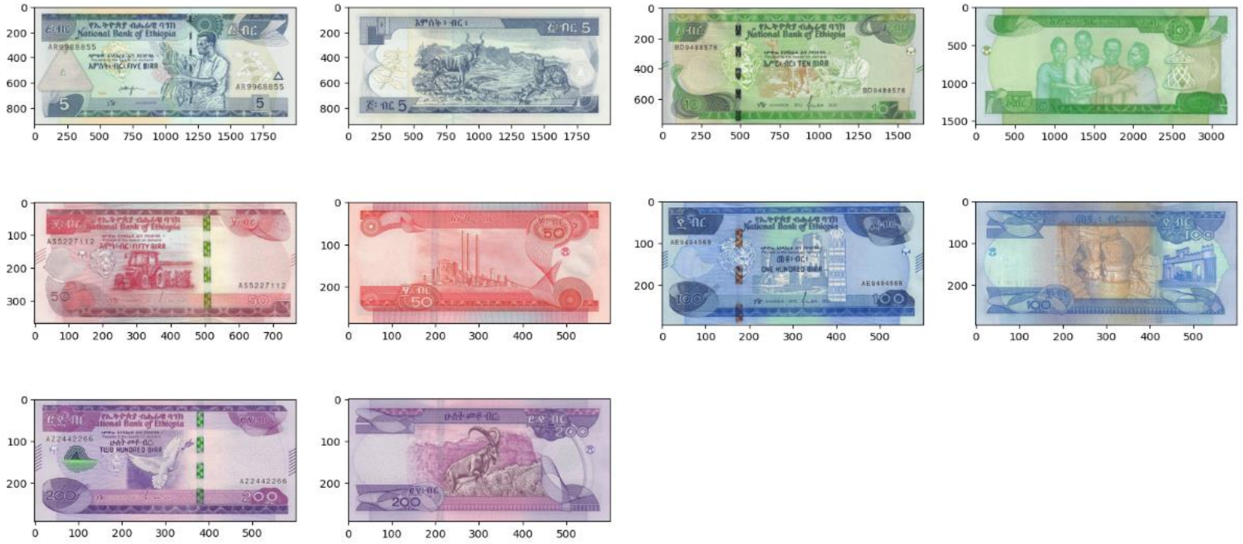


Figure 3- 2. Sample Genuine Ethiopian banknote

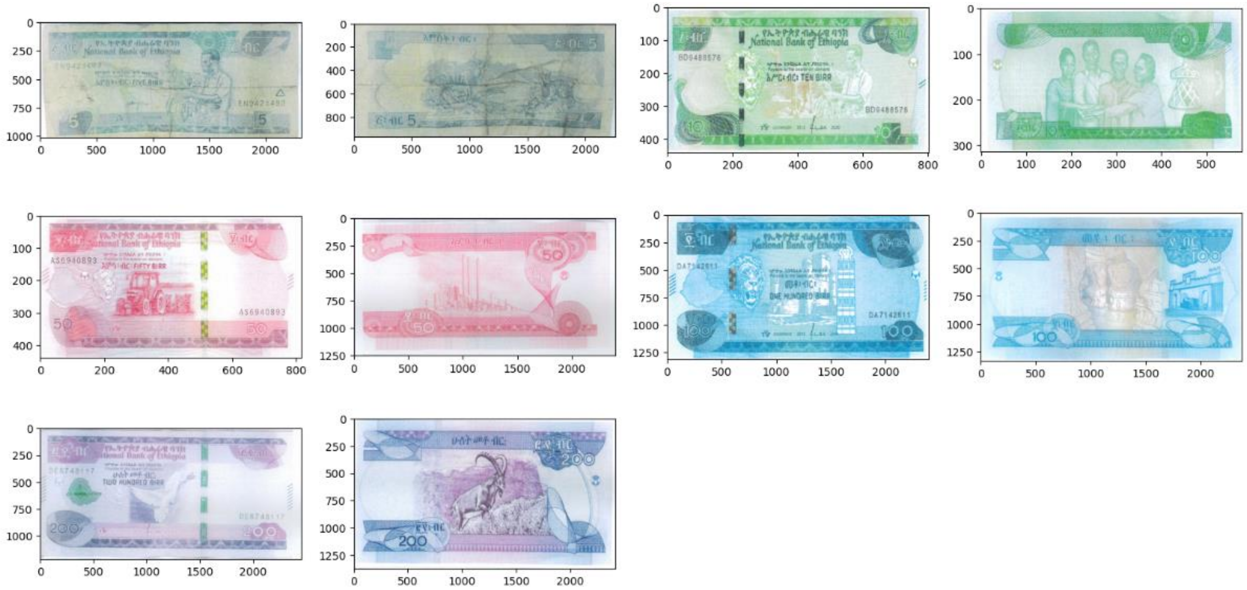
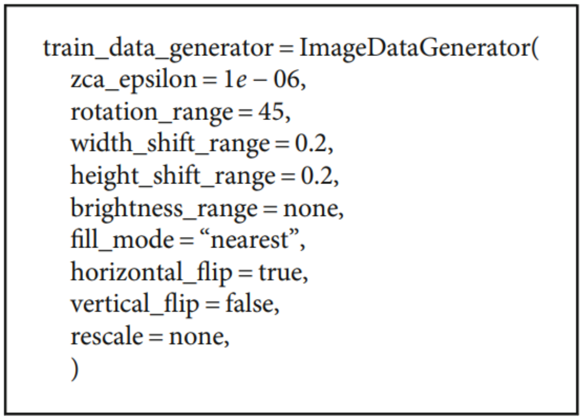


Figure 3- 3. Sample Fake Ethiopian banknote

Data Annotation: Once the banknote images have been collected, its need to be annotated with the correct labels to indicate whether they are genuine or counterfeit. This is done manually by annotators and using automated annotation tools. To guarantee that the model is trained on high-quality data, the annotations must be correct and consistent.

Data Cleaning and Preprocessing: The next step is to clean and preprocess the banknote images to remove noise, artifacts, and other distortions that may affect the performance of the model. We are uses various techniques such as image resizing, normalization, contrast adjustment, and color space conversion. Additionally, we use data augmentation techniques such as flipping, rotating, and cropping is used to increase the size and diversity of the dataset.



```
train_data_generator = ImageDataGenerator(
    zca_epsilon = 1e − 06,
    rotation_range = 45,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    brightness_range = none,
    fill_mode = "nearest",
    horizontal_flip = true,
    vertical_flip = false,
    rescale = none,
    )
```

Figure 3- 4. Pseudocode For data augmentation techniques

Data Splitting: Once the banknote images have been cleaned and preprocessed, we are split into training, validation, and testing datasets. The training set is used to train the model, while the validation set is used to fine-tune the hyper parameters of the model and prevent overfitting. The testing set is used to evaluate the performance of the model on unseen data.

Data Balancing: It is important to ensure that the dataset is balanced between genuine and counterfeit banknotes to prevent bias in the model's predictions.

Table 3- 1. Genuine Ethiopian Banknote Dataset

Genuine Banknote Dataset				
Type of Currency	Total	Training = 70%	Validation = 20%	Testing = 10%
Five / 5 birr	2500	1750	500	250
Ten / 10 birr	2500	1750	500	250
Fifty / 50 birr	2500	1750	500	250
Hundred / 100 birr	2500	1750	500	250
Two Hundred / 200 birr	2500	1750	500	250

Table 3- 2. Fake Ethiopian Banknote Dataset

Fake Banknote Dataset				
Type of Currency	Total	Training =70%	Validation = 20%	Testing = 10%
Five / 5 birr	2500	1750	500	250
Ten / 10 birr	2500	1750	500	250
Fifty / 50 birr	2500	1750	500	250
Hundred / 100 birr	2500	1750	500	250
Two Hundred / 200 birr	2500	1750	500	250

3.4 Pre-trained Model Selection

The CNN model has demonstrated impressive performance across various image processing tasks. Nonetheless, training such models from scratch to accurately classify counterfeit and genuine banknotes can be challenging due to the limited availability of counterfeit samples. To overcome this issue, transfer learning, a technique that leverages pre-trained models, has been employed in this study. Transfer learning offers several advantages, including mitigating the risk of overfitting when training with a small number of counterfeit banknote images, enhancing prediction accuracy, and reducing computational complexity during training. By utilizing transfer learning, the model benefits from the knowledge acquired from a larger dataset, leading to improved results in counterfeit banknote classification. In this study, four well-known pre-trained deep learning CNN InceptionV3 [38], VGG16 [39], MobileNet [40] and ResNet50 [41] were trained and tested to detect and classify banknotes.

Each model has its strengths and weaknesses, so we are trying to understand their architectures, performance on benchmark datasets, and suitability for this specific Ethiopian bank note counterfeit verification system.

Consider model performance: pre-trained models that have achieved high accuracy and good generalization on standard image classification datasets like ImageNet. Models that have performed well on diverse datasets are likely to have learned robust features that can be beneficial for this research.

InceptionV3: is a widely used CNN architecture known for its effectiveness in image recognition tasks. It utilizes inception modules and achieves high accuracy on ImageNet classification, achieving top-5 error rates below 4%. It has been successfully applied in various computer vision tasks.

VGG16: is a deep CNN architecture with a simple design that uses 3x3 convolutional filters. It has achieved excellent performance on various image classification benchmarks. And shown strong performance on ImageNet, achieving top-5 error rates below 8%. It provides a good balance between accuracy and simplicity.

MobileNet: is designed specifically for mobile and resource-constrained devices. It focuses on lightweight architecture with depth-wise separable convolutions, making it computationally efficient. Its performance might be slightly lower than deeper architectures like InceptionV3 or ResNet50, but it offers faster inference on resource-constrained devices.

ResNet50: is part of the ResNet family of CNN architectures. It introduced the concept of residual connections, enabling the training of very deep networks. ResNet50 has achieved outstanding accuracy and generalization.. It offers exceptional accuracy, especially for deep neural networks.

Complexity and computational requirements: Assess the complexity and computational requirements of the pre-trained models. Consider factors such as the number of layers, number of parameters, and memory usage. Models with fewer parameters might be faster to fine-tune and more suitable for deployment on resource-constrained systems.

Inception-v3: has a greater number of layers and parameters than some other models, which might increase computational requirements during fine-tuning and inference.

VGG16: has a relatively high number of layers and parameters, which can be computationally expensive, especially during training and inference.

MobileNet: has a lightweight architecture with fewer parameters, making it computationally efficient and suitable for resource-constrained environments.

ResNet50: has a deep architecture, but it introduces residual connections, allowing for effective training of deep networks without significantly increasing computational complexity.

Transfer learning capability: Ensure that the pre-trained model allows for transfer learning. Transfer learning allows to leverage the knowledge gained by the model during its pre-training on a large dataset. Check if the pre-trained model provides the flexibility to modify or replace the last few layers to adapt it to specific task.
Inception-v3, VGG16, MobileNet, and ResNet50 all support transfer learning and can modify or replace the last few layers to adapt them to specific Ethiopian currency detection and counterfeit verification task.

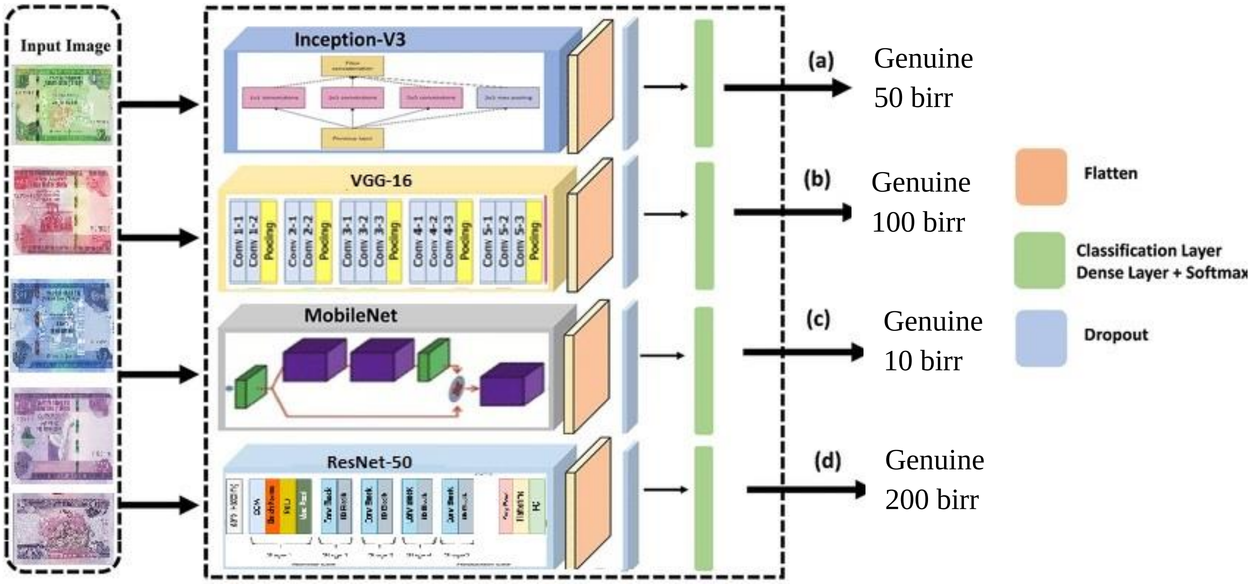


Figure 3- 5. Model Architecture Inception V3,. VGG16, MobileNet, and ResNet50

Domain similarity: Consider the domain similarity between the pre-trained model's training dataset and currency dataset. The more similar the domains, the better the pre-trained model can capture relevant features for similar task. However, even if the domains are not directly related, pre-trained models can still provide a good starting point due to their ability to learn generic visual features.

The pre-trained models (Inception-v3, VGG16, MobileNet, and ResNet50) have been trained on large-scale image classification tasks, which might not directly align with Ethiopian currency detection. However, their learned features can still capture useful visual representations that can be beneficial for this task.

Community support and resources: Check for community support and available resources related to the pre-trained models we are considering. Communities like TensorFlow and PyTorch often provide documentation, tutorials, and code examples that can help integrate and fine-tune the models effectively.

InceptionV3, VGG16, MobileNet, and ResNet50 have a significant amount of community support, available documentation, and resources in popular deep learning frameworks like TensorFlow and PyTorch.

Based on the evaluation, all four pre-trained models (InceptionV3, VGG16, MobileNet, and ResNet50) can be suitable choices for Ethiopian currency detection and counterfeit verification task. Further experiment with these models by fine-tuning and evaluating their performance using specific Ethiopian currency dataset to determine which one best meets accuracy requirements and computational constraints.

3.5 Fine-tuning the Model

Fine-tuning, as shown in Figure 3- 6, involves unfreezing the pre-trained base model (or a portion of it) and retraining it on new data using a relatively low learning rate. This process allows for gradual adaptation of the pre-trained features to the new data, leading to significant improvements in performance. Once the initial pre-trained model has converged to the new data, we can initiate the fine-tuning process by unfreezing all layers of the model and retraining it with a very low learning rate. It is important to note that since we are working with a larger model (including the frozen layers) and a small dataset, using a very low learning rate helps prevent overfitting. By incrementally adjusting the pre-trained weights with a conservative learning rate, we ensure that the updates to the weights are made gradually rather than making large adjustments rapidly.

One method to enhance performance involves simultaneously training (fine-tuning) the weights of the top layers of the pre-trained model while learning the newly added classifier. During this training process, the weights will be adjusted from generic feature maps to features that are specific to our dataset.

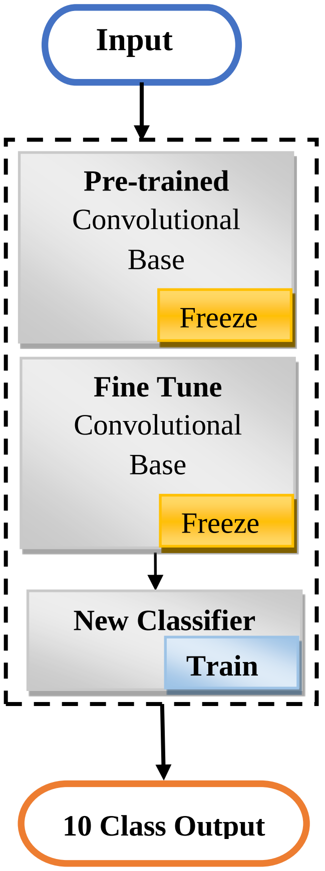
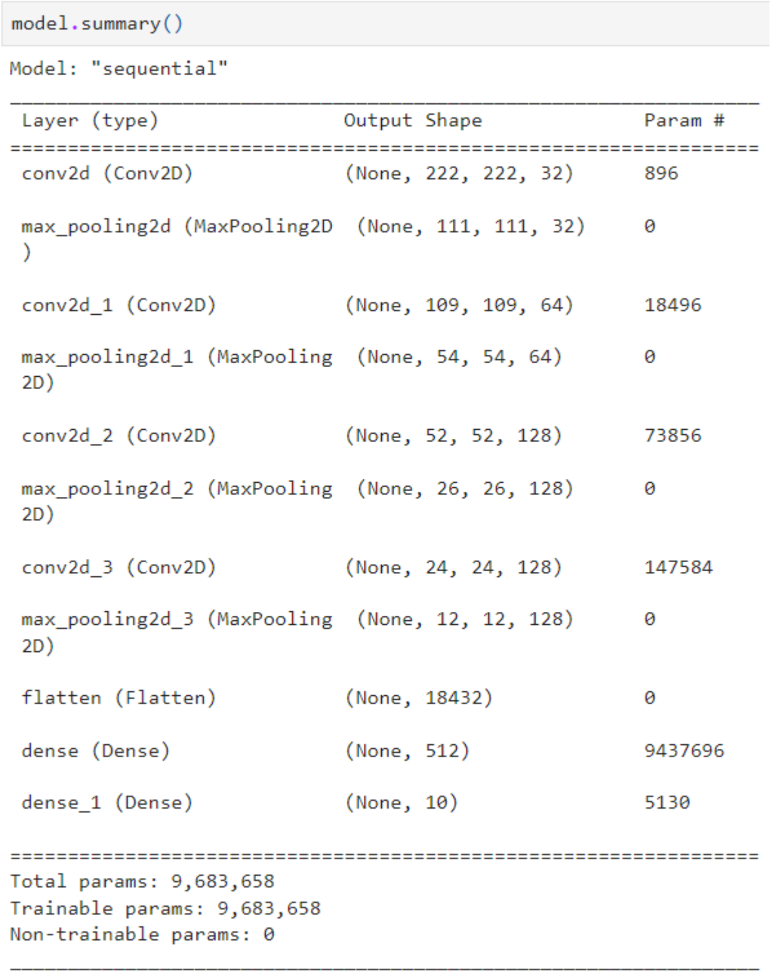


Figure 3- 6. Fine-tuning Method



```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 24, 24, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 128)      0
 2D)

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 512)               9437696

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 9,683,658
Trainable params: 9,683,658
Non-trainable params: 0
_________________________________________________________________
```

Figure 3- 7. Fine-tuned Model Summary

3.5.1 Classifier Network Building Blocks

Activation Function

The activation function plays a crucial role in a neural network as it transforms the weighted input of a node into its corresponding output or activation. One commonly used activation function is the rectified linear activation function, commonly known as ReLU. It is a piecewise linear function that outputs the input directly if it is positive, and zero otherwise. ReLU has gained popularity and is widely used as the default activation function in many neural network models. This is primarily due to its faster training speed and generally improved performance. In our specific model, we have employed ReLU as the chosen activation function. [42].

Pooling Layer

The pooling layer is a crucial component of our classifier network as it reduces the size of the feature map based on specified parameters. This helps in reducing the number of parameters and weights, resulting in faster training and lower computational costs. Various pooling methods exist, such as maximum pooling and average pooling. The pooling size and stride size have an impact on the output of the pooling process. In particular, global average pooling performs significant down sampling and is unique compared to other pooling methods. This down sampling operation is conducted before the fully connected layer and is typically applied only once in the model.

Fully Connected Layer

The final stage of the network architecture involves one or more fully connected layers, which are connected to the last convolutional, activation, or pooling layers. In these fully connected layers, each neuron is connected to all neurons in the preceding and succeeding layers. The purpose of the fully connected layer is to transform the two-dimensional feature map into a one-dimensional feature vector. This layer captures high-level features that are strongly associated with each class. In our specific case, the number of neurons in the final fully connected layer corresponds to the number of classes, which is 10.

Dropout Layer

To mitigate the issue of overfitting in our model, we employ the technique of dropout. Specifically, we incorporate dropout between two fully connected layers with a dropout rate of 0.45. Dropout involves randomly disconnecting inputs from the first fully connected layer to the second fully connected layer. By introducing this dropout mechanism, additional nodes are activated in a redundant manner when presented with similar patterns. This helps our model to generalize better and effectively reduce overfitting.

Softmax Layer

The choice of activation function in the final fully connected layer depends on the task at hand. For multi-class classification, we utilize the softmax function as a classifier. The softmax function enables the calculation of probabilities for multiple classes simultaneously. It adjusts the probability of each target class based on the real values generated from the previous fully connected layer. Consequently, the output of the softmax function represents a probability distribution, where values range from 0 to 1 and the sum of all values equals 1. In our case, the softmax layer has ten nodes, corresponding to the number of classes in our classification task.

Loss Function

The loss function is a crucial component of CNNs, as it quantifies the model's prediction error and guides the calculation of gradients for updating the network's parameters. The specific loss function used determines how the loss is computed. In our approach, we utilized categorical cross-entropy, which is commonly used for multi-class classification tasks. This loss function allows us to measure the discrepancy between the predicted probabilities and the true class labels, enabling effective training of the model.

Optimizers

Optimization techniques are employed to minimize the error function in neural networks. These techniques involve adjusting the learnable parameters, such as weights and biases, to compute the output values. Optimizers play a crucial role in the training process as they aid in reducing the loss. There are various types of optimizers available, and researchers often select a specific optimizer for their experiments. In our case, we have chosen the Adam optimizer based on previous research findings and its effectiveness in optimizing neural network models.

3.6 Model Evaluation

3.6.1 Model Parameters

The model has different parameters that must be selected carefully in order to get a better accurate model and to minimize losses, here the following are some parameters of the model:

Epoch: is a hyper parameter that determines the number of complete passes made by the machine learning algorithm over the entire training dataset or the number of times the neural network is trained using all the training data within a single cycle. During each epoch, all the data is utilized precisely once, and both the forward pass and backward pass within the epoch are considered as a single pass. Our model is trained using an epoch size of 100.

Batch size: is a hyper parameter that determines the number of samples processed before updating the internal model parameters. It is utilized to reduce the computational load and memory requirements by working with a smaller subset of samples. By using a smaller batch size, the network operates on fewer samples at a time, resulting in reduced memory usage during the training process. In our model, we have employed a batch size of 32 consistently.

Learning Rate: The learning rate is a configurable hyper parameter used during the training of neural networks. It is a small positive value typically ranging between 0.0 and 1.0. The learning rate plays a crucial role in determining the speed at which the model adapts to the problem at hand. It acts as a tuning parameter in the optimization algorithm, dictating the step size taken at each iteration as the model progresses towards minimizing the loss function.

If the learning rate is set too low, the training process progresses slowly. Conversely, if the learning rate is set too high, it can lead to undesirable divergent behavior in the loss function.

In this study, we compare different optimizers, such as SGD and ADAM, by evaluating their performance using the dataset. The goal is to select the best optimizer. After testing the model, we found that the SGD optimizer with a learning rate of 0.0001 produced superior results compared to the ADAM optimizer.

3.6.2 Performance Evaluation Metrics

To assess the effectiveness of our classification model, we rely on various metrics that gauge its performance. These metrics serve as evaluation measures to determine the model's competence. Accuracy, Recall, F1-score, precision, and confusion matrices are among the most commonly employed metrics used to assess our model's performance. Furthermore, we also consider the training time as an evaluation criterion for the model.

Classification Accuracy

Accuracy is a metric utilized in the evaluation of classification models. It represents the proportion of correct predictions made by our model, often expressed as a percentage. Accuracy reflects the closeness of predicted values to the actual values within the test data. The mathematical expression for accuracy is calculated as follows:

The accuracy of a classification can also be derived from the confusion matrix generated by the model. The mathematical expression used to calculate accuracy based on the confusion matrix is:

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

Where

TP – True Positive

TN – True negative

FP – False Positive

FN – False Negative

Precision

Precision is the ratio of true positives to all positive predicted by the model and it enables to measure how precise the classifier when it predicts the positive instances.

When the model has more false positives predicts, then it will have lower precision. The precision is calculated as the following:

$$Precision\ = \frac{Number\ of\ Correct\ postive\ prediction}{Total\ number\ of\ Postive\ prediction}$$

We can also calculate precision from the confusion matrix results by using the following formula

$$Precision\ = \frac{TP}{TP + FN}$$

Recall

It is metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made and mathematically it is expressed as:

$$Recall = \frac{TP}{TP\ +\ FN}$$

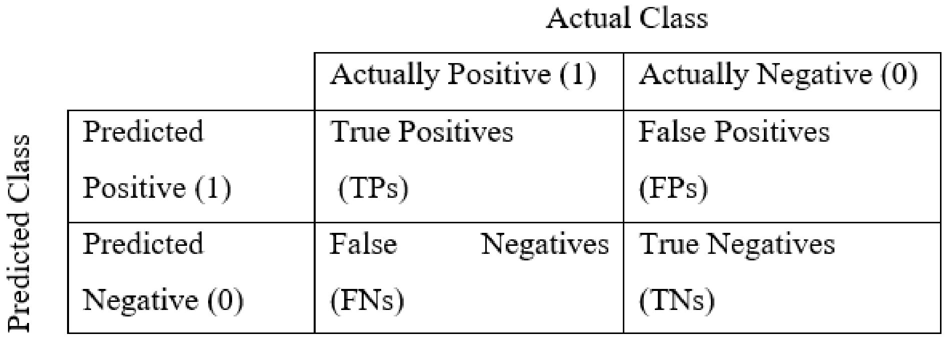
F1 score

It is the weighted average of Precision and Recall mathematically it is expressed as the following:

$$F1\ score = \frac{2\ *\ (Precision\ *\ Recall)}{Precision\ +\ Recall}$$

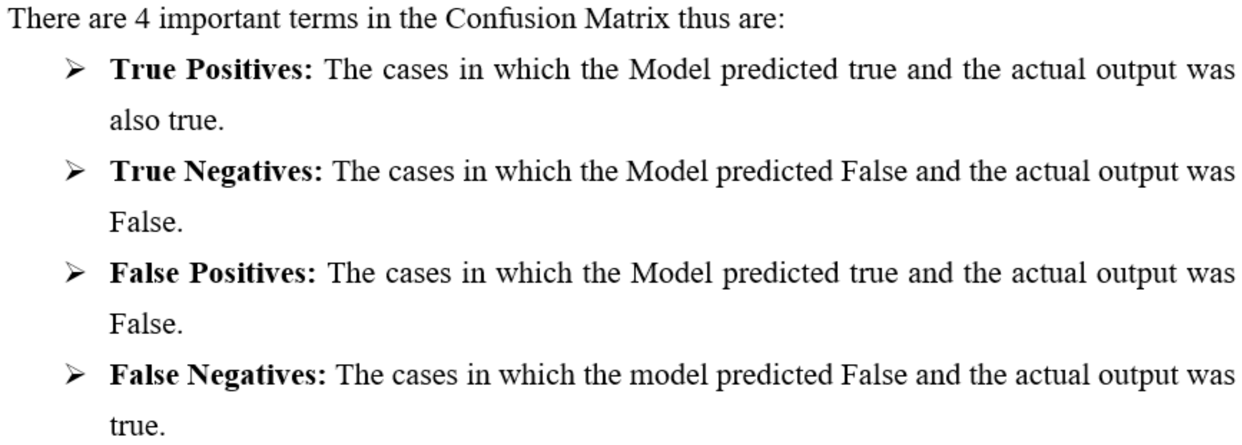
Confusion Matrix

The confusion matrix is a performance evaluation metric used to assess a classification model. It involves counting the number of test records that are correctly and incorrectly predicted by the model. The confusion matrix offers valuable insights into the model's performance by illustrating how it becomes confused or makes errors in classifying instances. It provides a comprehensive view of the model's performance, including the accuracy of predictions for different classes, the types of correct and incorrect predictions made, and the overall performance of the model. The confusion matrix presents its results in a matrix format, offering a comprehensive description of the model's performance.



		Actual Class	
		Actually Positive (1)	Actually Negative (0)
Predicted Class	Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
	Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 3- 8. Confusion matrix



There are 4 important terms in the Confusion Matrix thus are:

- **True Positives:** The cases in which the Model predicted true and the actual output was also true.
- **True Negatives:** The cases in which the Model predicted False and the actual output was False.
- **False Positives:** The cases in which the Model predicted true and the actual output was False.
- **False Negatives:** The cases in which the model predicted False and the actual output was true.

3.7 Developing Tools

This study is about developing the model for Ethiopian banknote classification and counterfeit verification. The implementation of the study is performing by using the following tools:

- **Python Programming language**: a most power full high level programming language that supports different modules, to evaluate mathematical operations also.

- **Tensor flow**: an end to end open-source machine learning platform used for numerical computations and it is a symbolic math library based on the data flow and the programming that enables also to run efficiently on the central processing unit (CPU) or graphical processing unit (GPU)

- **Keras**: is a deep learning framework and a high-level interface and it is a python API that runs on other libraries or platforms like tensor flow and it supports most modules of the neural network so it enables python interface with neural network.

- The research is undertaken on Python, operated on the Microsoft Windows 11 64-bit operating system of Hp Pavilion laptop. The laptop embeds Intel(R) Core (TM) i7-4710MQ CPU @ 2.60GHZ processor, Graphics Adapter: NVIDIA GeForce MX450, with 8 GB RAM.

- The actual training, testing and validation is done by Dell Server embeds Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz, 2694 MHz, 28 Core(s), 56 Logical Processers, Graphics Adapter: NVIDIA Quadro RTX 6000 GPU, with 192 GB RAM

3.8 Chapter Summary

This chapter provides a detailed description of the system architecture, dataset preparation, model design, hardware, and software specifications utilized in this study. The section on model design introduces the overall architecture and various components of the model. Additionally, it presents a comprehensive overview of the evaluation metrics employed to measure the performance of the model.

CHAPTER FOUR

4. RESULT AND DISCUSSION

4.1 Overview

This chapter presents the implementation details of the developed model and provides an overview of the experimental results obtained. The primary objective of this study was to create a model specifically designed for Ethiopian banknote classification and counterfeit verification. The experimental results section encompasses various stages, including image preprocessing, model training, and model testing. The implementation of the model was carried out using the Python programming language with the TensorFlow framework and Keras library.

The model architecture consists of five convolutional layers followed by Max pooling layers, all utilizing the ReLU activation function. It also includes fully connected layers. The output layer employs the SoftMax activation function to classify the output image into ten distinct classes.

4.2 Dataset

The dataset used in this study consists of ten classes, namely: Genuine 5-birr notes, Genuine 10-birr notes, Genuine 50-birr notes, Genuine 100-birr notes, Genuine 200-birr notes, Fake 5-birr notes, Fake 10-birr notes, Fake 50-birr notes, Fake 100-birr notes, and Fake 200-birr notes. Each class contains 250 images, resulting in a total of 2500 images.

To address class imbalance and increase the dataset size, data augmentation techniques were applied. This approach helps balance the number of datasets in each class and enhances the overall dataset volume.

The dataset was partitioned into three sets: 70% for training, 20% for testing, and 10% for validation. The images within the dataset were saved in JPG format and had a pixel size of 224 by 224.

4.3 Implementation Tools

The experiments in this study were done based on the prototype developed with Keras (Tensor Flow as a backend) on Intel Core ™ i5-5200 CPU, and 8 GB of RAM. The datasets for the study were partitioned into 70 percent for training, 20 percent for testing, and 10 percent for validation.

The model is trained with learning rate of 0.0001 for 100 epochs. We have used precision, recall, and f1-score, for measuring the performance of our model. In addition, we have also used macro-average, weighted average, and confusion matrices for measuring the model performance.

4.5 Experimental Results

We have done different experiments in model training, model testing and also for comparison of the model with the state-of-the-art models. The training phase of the model includes the consecutive sequences of convolution, pooling, activation function, dropout, fully-connected layers, and SoftMax classifier for classification of defects. In training phase, we made comparison of optimizers for selecting the better optimizer by training them with different learning rates. We conducted experiments on fully connected CNN, and with state-of-the-art on InceptionV3, VGG16, MobileNet, and ResNet50 pre-trained models. The performance evaluation of the experiments has been done by using the performance metrics described on the above section.

Table 4- 1. Model’s parameters configuration

Metrics	**Value**
Input Shape	224×224
Shuffling	Each epoch
Bach size	32
Learning Rate	0.0001
Epoch	100
Flip	Horizontal Vertical
Activation	Softmax (Final Classification Layer)
Optimizer	Adam, SGD

4.5.1 Counterfeit detection using Fine-tuned InceptionV3

InceptionV3 is a model with 92MB size and 23M parameters. The proposed model was fine-tuned using our dataset and subsequently tested. The training process was carried out for a maximum of 100 epochs. Below, we present the commonly used classification metrics and training curves for evaluating the performance of the model.

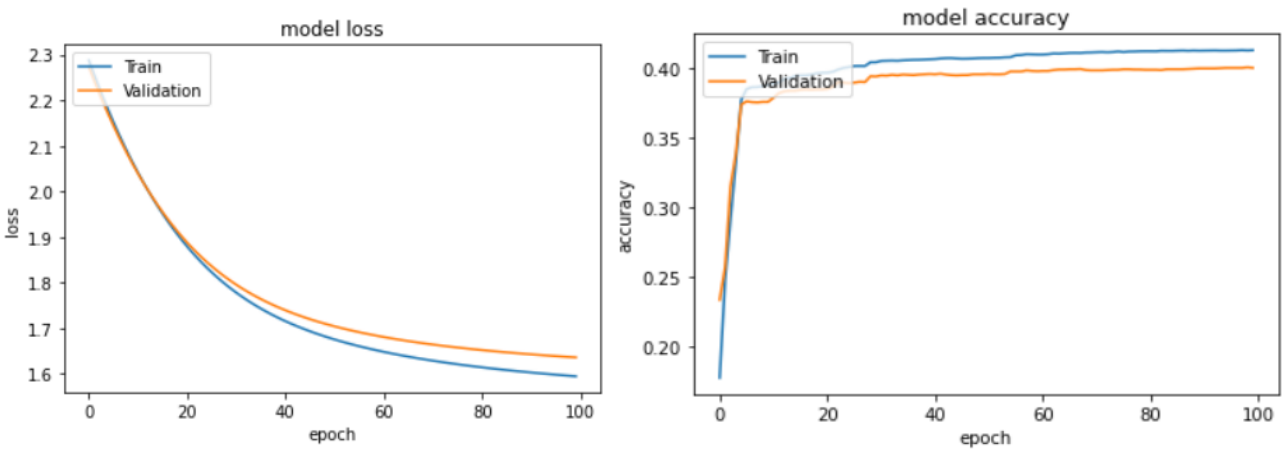
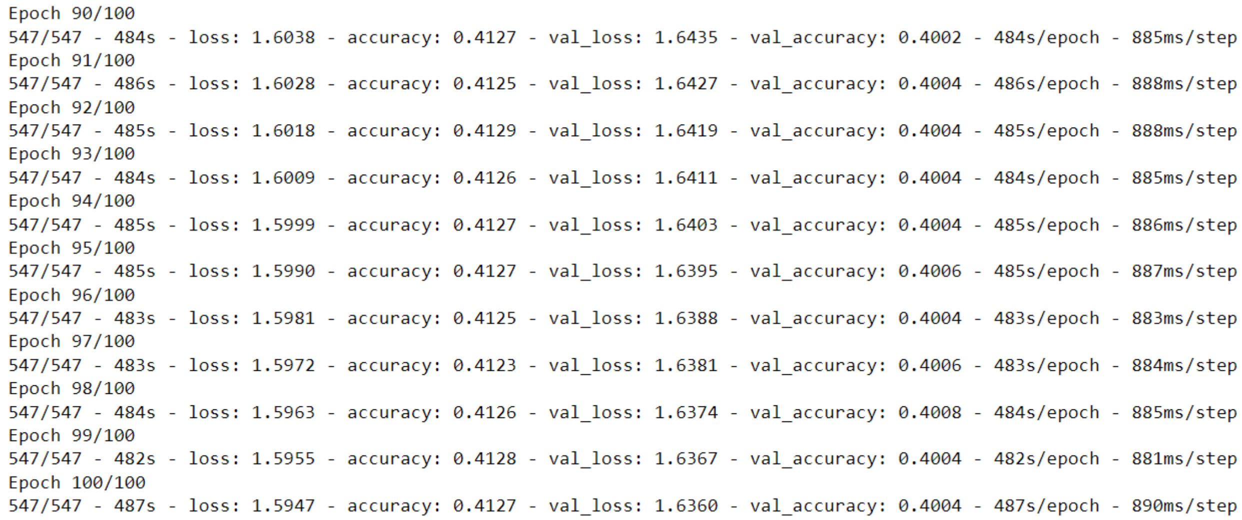


Figure 4- 1. InceptionV3 loss and accuracy curves

Based on the depicted loss and accuracy curves, it can be observed that at epoch 100 with a learning rate of 0.0001 (which is notably low), the validation loss was recorded as 1.63. At epoch 22 with the same learning rate, the last recorded validation accuracy was 40.0%, while the training accuracy reached 41.2%.



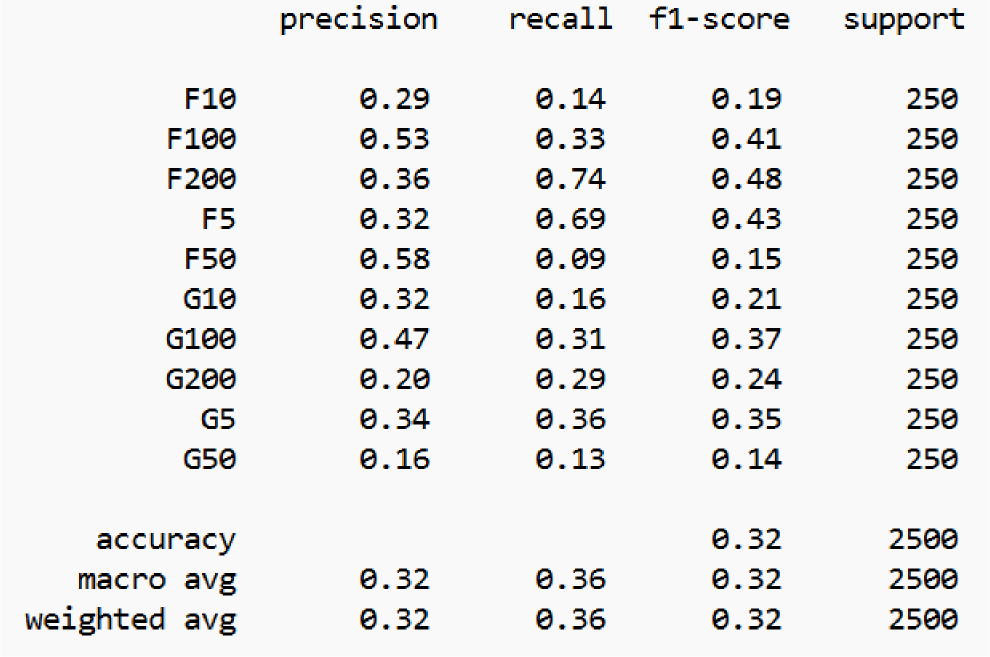
```
Epoch 90/100
547/547 - 484s - loss: 1.6038 - accuracy: 0.4127 - val_loss: 1.6435 - val_accuracy: 0.4002 - 484s/epoch - 885ms/step
Epoch 91/100
547/547 - 486s - loss: 1.6028 - accuracy: 0.4125 - val_loss: 1.6427 - val_accuracy: 0.4004 - 486s/epoch - 888ms/step
Epoch 92/100
547/547 - 485s - loss: 1.6018 - accuracy: 0.4129 - val_loss: 1.6419 - val_accuracy: 0.4004 - 485s/epoch - 888ms/step
Epoch 93/100
547/547 - 484s - loss: 1.6009 - accuracy: 0.4126 - val_loss: 1.6411 - val_accuracy: 0.4004 - 484s/epoch - 885ms/step
Epoch 94/100
547/547 - 485s - loss: 1.5999 - accuracy: 0.4127 - val_loss: 1.6403 - val_accuracy: 0.4004 - 485s/epoch - 886ms/step
Epoch 95/100
547/547 - 485s - loss: 1.5990 - accuracy: 0.4127 - val_loss: 1.6395 - val_accuracy: 0.4006 - 485s/epoch - 887ms/step
Epoch 96/100
547/547 - 483s - loss: 1.5981 - accuracy: 0.4125 - val_loss: 1.6388 - val_accuracy: 0.4004 - 483s/epoch - 883ms/step
Epoch 97/100
547/547 - 483s - loss: 1.5972 - accuracy: 0.4123 - val_loss: 1.6381 - val_accuracy: 0.4006 - 483s/epoch - 884ms/step
Epoch 98/100
547/547 - 484s - loss: 1.5963 - accuracy: 0.4126 - val_loss: 1.6374 - val_accuracy: 0.4008 - 484s/epoch - 885ms/step
Epoch 99/100
547/547 - 482s - loss: 1.5955 - accuracy: 0.4128 - val_loss: 1.6367 - val_accuracy: 0.4004 - 482s/epoch - 881ms/step
Epoch 100/100
547/547 - 487s - loss: 1.5947 - accuracy: 0.4127 - val_loss: 1.6360 - val_accuracy: 0.4004 - 487s/epoch - 890ms/step
```

Figure 4- 2. Training performance of InceptionV3 at the last 10 epochs

The presented output illustrates the loss and accuracy metrics for both the training and validation sets. The loss metric gauges the model's degree of fit to the training data, while accuracy measures its ability to correctly predict labels for the training data. The val_loss and val_accuracy metrics pertain to the validation set, representing the same evaluation measures.

The model underwent training for a duration of 487 seconds, which corresponds to approximately 8 minutes. During this training period, the model attained an accuracy of 41.27% on the training set and an accuracy of 40.00% on the validation set.

However, it is worth noting that the model did not achieve satisfactory accuracy levels on either the training or validation set. This suggests that the model may be overfitting to the training data, resulting in limited generalization to unseen data.



```
              precision    recall  f1-score   support

         F10       0.29      0.14      0.19       250
        F100       0.53      0.33      0.41       250
        F200       0.36      0.74      0.48       250
          F5       0.32      0.69      0.43       250
         F50       0.58      0.09      0.15       250
         G10       0.32      0.16      0.21       250
        G100       0.47      0.31      0.37       250
        G200       0.20      0.29      0.24       250
          G5       0.34      0.36      0.35       250
         G50       0.16      0.13      0.14       250

    accuracy                           0.32      2500
   macro avg       0.32      0.36      0.32      2500
weighted avg       0.32      0.36      0.32      2500
```

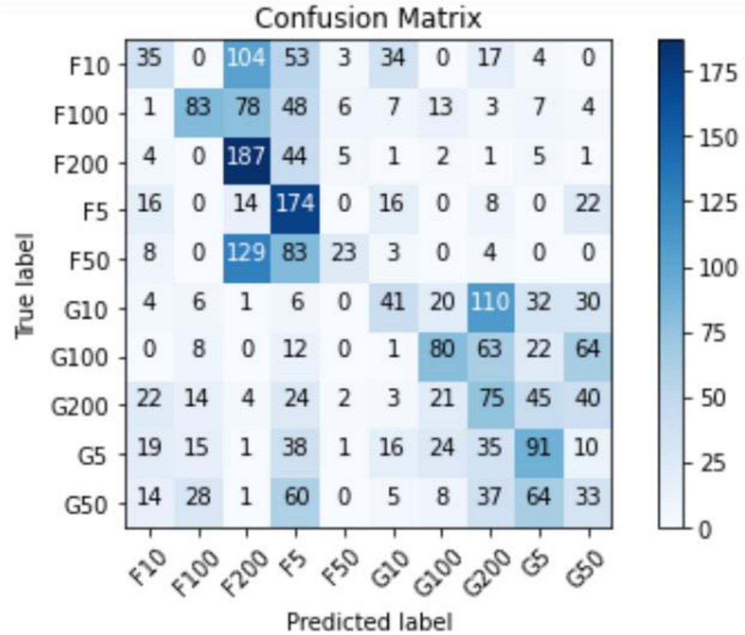
Figure 4- 3. Test Set Classification report using fine-tuned InceptionV3

The precision is the percentage of predicted positive instances that are actually positive. The recall is the percentage of actual positive instances that are predicted positive. The F1-score is a harmonic mean of the precision and recall.

The accuracy is the percentage of instances that are correctly classified. The macro average precision, recall, and F1-score are calculated by averaging the precision, recall, and F1-score for each class, without taking into account the class imbalance. The weighted average precision, recall, and F1-score are calculated by averaging the precision, recall, and F1-score for each class, weighting each class by its number of instances.

In this case, the model is not very accurate. The precision, recall, and F1-score for each class are all low. This suggests that the model is not very good at identifying positive instances. The accuracy is also low, suggesting that the model is not very good at classifying instances overall.

For detailed and clear evaluation result for each class and see the rate of error distribution and perfect classification, the confusion matrix was used, as provided below:



	F10	F100	F200	F5	F50	G10	G100	G200	G5	G50
F10	35	0	104	53	3	34	0	17	4	0
F100	1	83	78	48	6	7	13	3	7	4
F200	4	0	187	44	5	1	2	1	5	1
F5	16	0	14	174	0	16	0	8	0	22
F50	8	0	129	83	23	3	0	4	0	0
G10	4	6	1	6	0	41	20	110	32	30
G100	0	8	0	12	0	1	80	63	22	64
G200	22	14	4	24	2	3	21	75	45	40
G5	19	15	1	38	1	16	24	35	91	10
G50	14	28	1	60	0	5	8	37	64	33

Figure 4- 4. Confusion matrix for Test Set classification with fine-tuned InceptionV3

The confusion matrix reveals that the class F50 exhibited the highest rate of misclassifications. Specifically, some images in this class were wrongly classified as either F5 or F200. On the other hand, the class F200 displayed the lowest rate of misclassifications. This observation suggests that the model faces difficulties in distinguishing certain features between classes, leading to confusion in accurately classifying some images.

4.5.2 Counterfeit detection using Fine-tuned VGG16

The VGG16 model, which has a size of 500MB and comprises 138 million parameters, was selected as the proposed model for fine-tuning the classifier. The training process was conducted for a maximum of 100 epochs. Below, we present various classification metrics and the loss versus accuracy curves to evaluate the performance of the model.

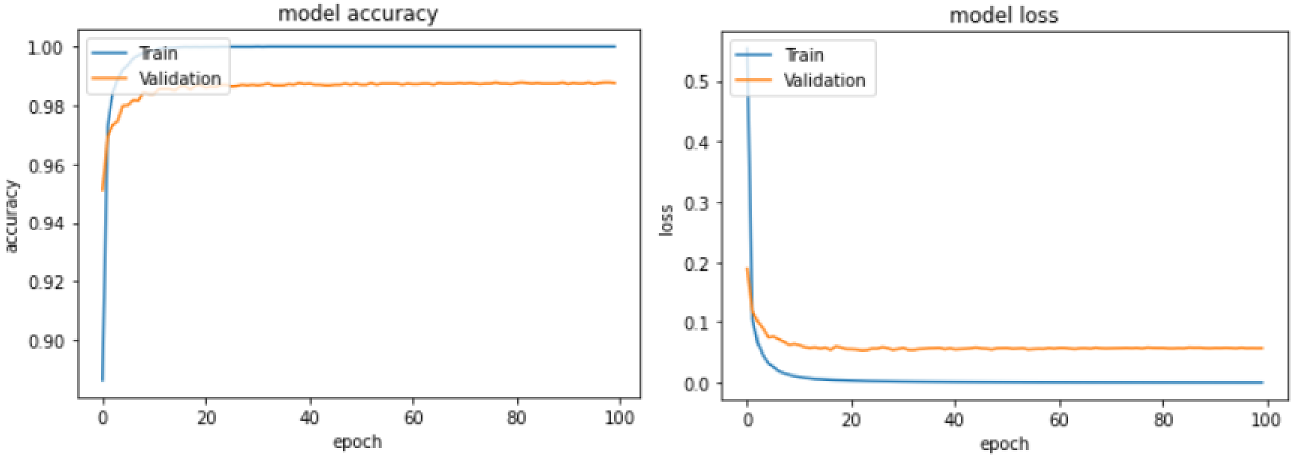
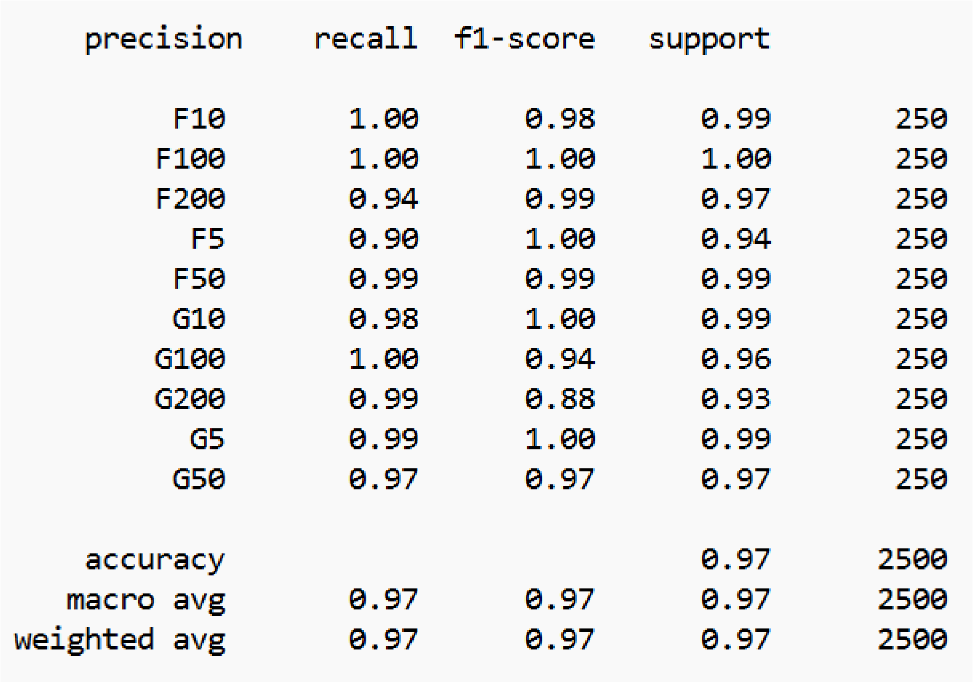


Figure 4- 5. VGG16 loss and accuracy curves

Based on the depicted loss and accuracy curves, it can be observed that at epoch 100 with a learning rate of 0.0001 (which is notably low), the validation loss was recorded as 0.21. At the same epoch and learning rate, the last recorded validation accuracy was 98.9%, while the training accuracy reached 99.7%.



```
              precision    recall  f1-score   support

         F10       1.00      0.98      0.99       250
        F100       1.00      1.00      1.00       250
        F200       0.94      0.99      0.97       250
          F5       0.90      1.00      0.94       250
         F50       0.99      0.99      0.99       250
         G10       0.98      1.00      0.99       250
        G100       1.00      0.94      0.96       250
        G200       0.99      0.88      0.93       250
          G5       0.99      1.00      0.99       250
         G50       0.97      0.97      0.97       250

    accuracy                           0.97      2500
   macro avg       0.97      0.97      0.97      2500
weighted avg       0.97      0.97      0.97      2500
```

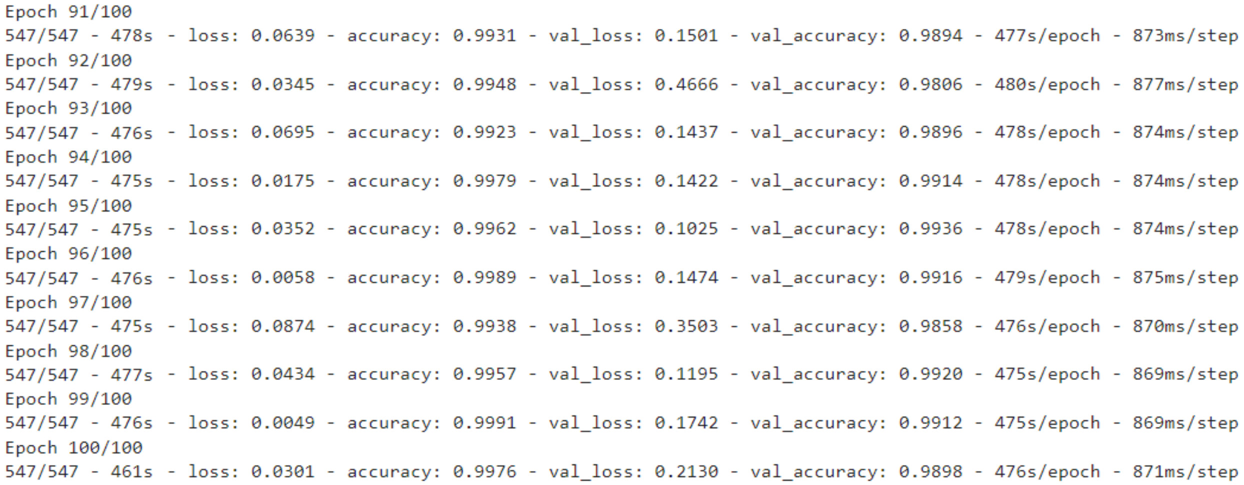
Figure 4- 6. Test Set counterfeit detection report using fine-tuned VGG16

The precision, recall, and F1-score for each class are shown in the table above. The accuracy is 97%, and the macro average precision, recall, and F1-score are all 97%. The weighted average precision, recall, and F1-score are also all 97%.

The precision is the percentage of predicted positive instances that are actually positive. The recall is the percentage of actual positive instances that are predicted positive. The F1-score is a harmonic mean of the precision and recall.

The accuracy is the percentage of instances that are correctly classified. The macro average precision, recall, and F1-score are calculated by averaging the precision, recall, and F1-score for each class, without taking into account the class imbalance. The weighted average precision, recall, and F1-score are calculated by averaging the precision, recall, and F1-score for each class, weighting each class by its number of instances.

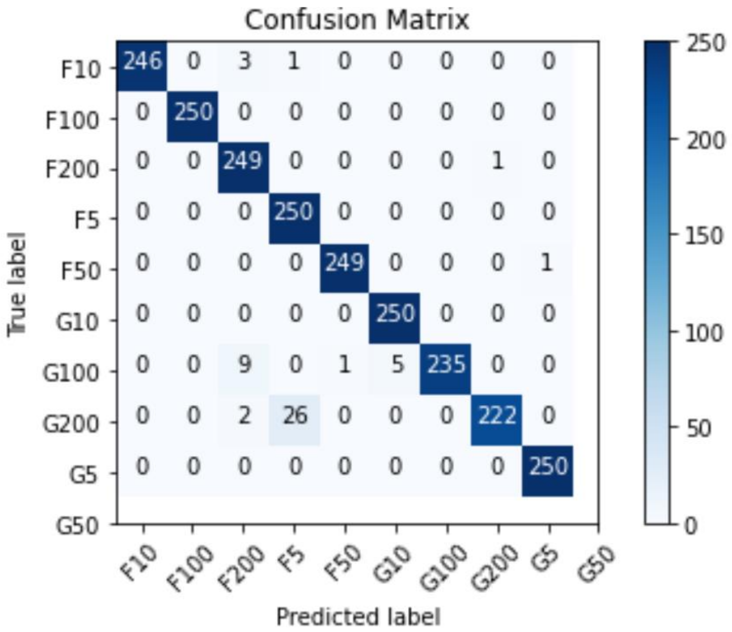
In this case, the model is very accurate. The precision, recall, and F1-score for each class are all high. This suggests that the model is very good at identifying positive instances. The accuracy is also high, suggesting that the model is very good at classifying instances overall.



```
Epoch 91/100
547/547 - 478s - loss: 0.0639 - accuracy: 0.9931 - val_loss: 0.1501 - val_accuracy: 0.9894 - 477s/epoch - 873ms/step
Epoch 92/100
547/547 - 479s - loss: 0.0345 - accuracy: 0.9948 - val_loss: 0.4666 - val_accuracy: 0.9806 - 480s/epoch - 877ms/step
Epoch 93/100
547/547 - 476s - loss: 0.0695 - accuracy: 0.9923 - val_loss: 0.1437 - val_accuracy: 0.9896 - 478s/epoch - 874ms/step
Epoch 94/100
547/547 - 475s - loss: 0.0175 - accuracy: 0.9979 - val_loss: 0.1422 - val_accuracy: 0.9914 - 478s/epoch - 874ms/step
Epoch 95/100
547/547 - 475s - loss: 0.0352 - accuracy: 0.9962 - val_loss: 0.1025 - val_accuracy: 0.9936 - 478s/epoch - 874ms/step
Epoch 96/100
547/547 - 476s - loss: 0.0058 - accuracy: 0.9989 - val_loss: 0.1474 - val_accuracy: 0.9916 - 479s/epoch - 875ms/step
Epoch 97/100
547/547 - 475s - loss: 0.0874 - accuracy: 0.9938 - val_loss: 0.3503 - val_accuracy: 0.9858 - 476s/epoch - 870ms/step
Epoch 98/100
547/547 - 477s - loss: 0.0434 - accuracy: 0.9957 - val_loss: 0.1195 - val_accuracy: 0.9920 - 475s/epoch - 869ms/step
Epoch 99/100
547/547 - 476s - loss: 0.0049 - accuracy: 0.9991 - val_loss: 0.1742 - val_accuracy: 0.9912 - 475s/epoch - 869ms/step
Epoch 100/100
547/547 - 461s - loss: 0.0301 - accuracy: 0.9976 - val_loss: 0.2130 - val_accuracy: 0.9898 - 476s/epoch - 871ms/step
```

Figure 4- 7. Training performance of VGG16 at the last 10 epochs

As clearly depicted in figure 16, the fine-tuned model scores 99 % training accuracy and 98% validation accuracy on the datasets. The training time of the model was 476 seconds per epoch (average training time per epoch). The model scores 99% classification accuracy. It shows that the fine-tuned model has performance improvement in total accuracy. For detailed and clear evaluation result in each class and to see the rate of misclassification we provide the confusion matrix as below.



True label \ Predicted label	F10	F100	F200	F5	F50	G10	G100	G200	G5	G50
F10	246	0	3	1	0	0	0	0	0	
F100	0	250	0	0	0	0	0	0	0	
F200	0	0	249	0	0	0	0	1	0	
F5	0	0	0	250	0	0	0	0	0	
F50	0	0	0	0	249	0	0	0	1	
G10	0	0	0	0	0	250	0	0	0	
G100	0	0	9	0	1	5	235	0	0	
G200	0	0	2	26	0	0	0	222	0	
G5	0	0	0	0	0	0	0	0	250	
G50										

Figure 4- 8. Confusion matrix for Test Set counterfeit verification with fine-tuned VGG16

Based on the analysis of the confusion matrix, it can be inferred that the class G200 classification exhibited the highest rate of misclassifications. Specifically, some data points were incorrectly classified as F5. On the other hand, the classes F100, F5, G10, and G5 displayed the lowest rates of misclassifications. This observation suggests that the model faces some challenges in accurately differentiating certain features between the G200 and F5 classes to some extent.

4.5.3 Classification using Fine-tuned MobileNet

The proposed model utilizes the MobileNet pre-trained model for training, validation, and testing. Among the different variations of MobileNet, MobileNet was chosen due to its high accuracy performance on the ImageNet dataset. Since we are fine-tuning this model for our specific problem with 10 classes, the top 4 layers of the pre-trained MobileNet architecture were modified and fine-tuned to ensure optimal performance for our problem.

The MobileNet model was trained for a maximum of 100 epochs, and the following results were obtained from this training process.

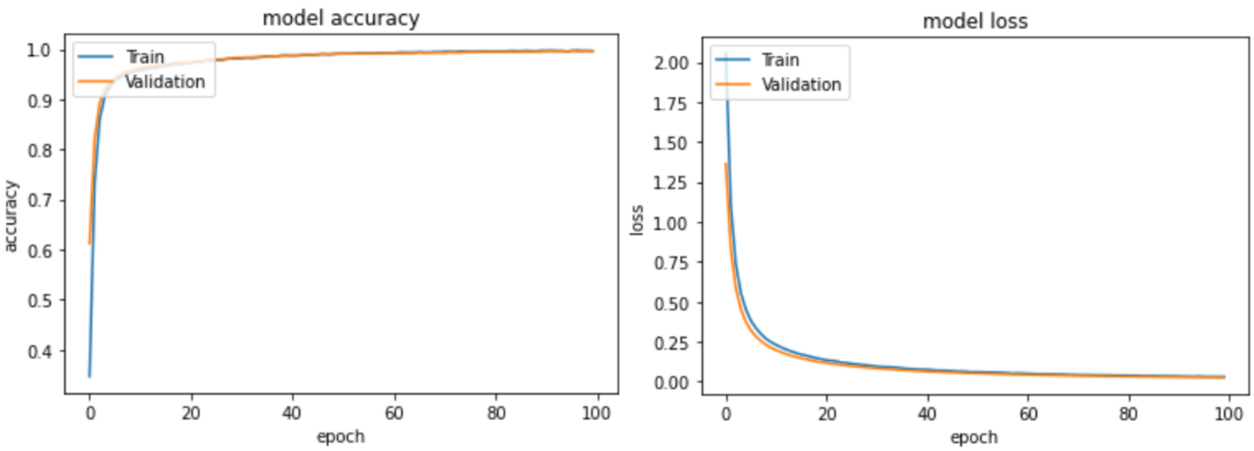
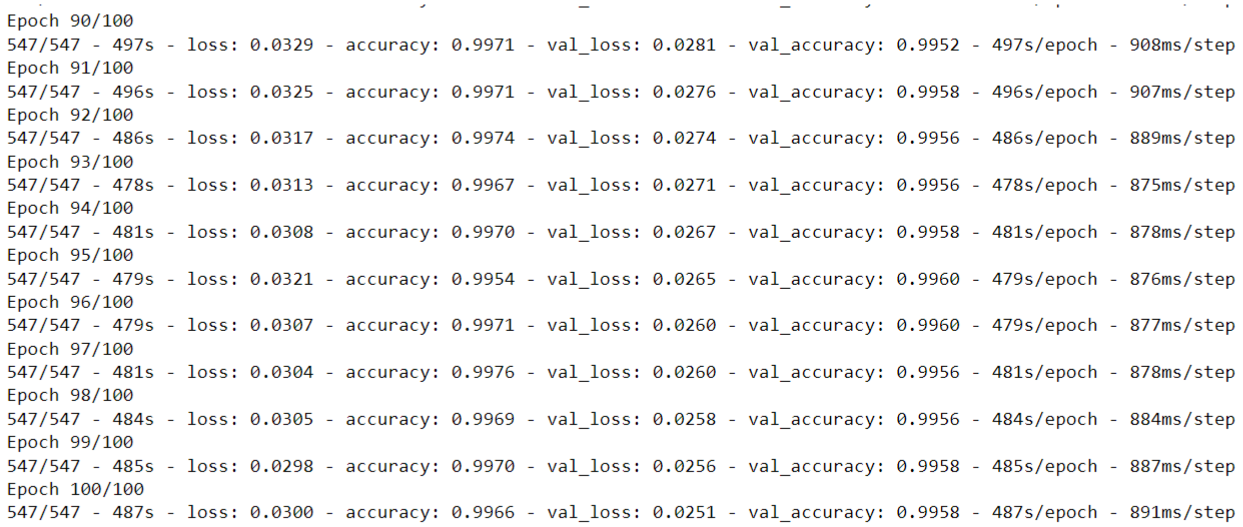


Figure 4- 9. MobileNet training loss and accuracy curves

By examining the training loss and accuracy curves, it is evident that at epoch 100 with a learning rate of 0.0001 (which is notably low), the last recorded validation loss was 0.03. Simultaneously, the highest validation accuracy of 99.58% and training accuracy of 99.66% were achieved at epoch 100 with the same learning rate.



```
Epoch 90/100
547/547 - 497s - loss: 0.0329 - accuracy: 0.9971 - val_loss: 0.0281 - val_accuracy: 0.9952 - 497s/epoch - 908ms/step
Epoch 91/100
547/547 - 496s - loss: 0.0325 - accuracy: 0.9971 - val_loss: 0.0276 - val_accuracy: 0.9958 - 496s/epoch - 907ms/step
Epoch 92/100
547/547 - 486s - loss: 0.0317 - accuracy: 0.9974 - val_loss: 0.0274 - val_accuracy: 0.9956 - 486s/epoch - 889ms/step
Epoch 93/100
547/547 - 478s - loss: 0.0313 - accuracy: 0.9967 - val_loss: 0.0271 - val_accuracy: 0.9956 - 478s/epoch - 875ms/step
Epoch 94/100
547/547 - 481s - loss: 0.0308 - accuracy: 0.9970 - val_loss: 0.0267 - val_accuracy: 0.9958 - 481s/epoch - 878ms/step
Epoch 95/100
547/547 - 479s - loss: 0.0321 - accuracy: 0.9954 - val_loss: 0.0265 - val_accuracy: 0.9960 - 479s/epoch - 876ms/step
Epoch 96/100
547/547 - 479s - loss: 0.0307 - accuracy: 0.9971 - val_loss: 0.0260 - val_accuracy: 0.9960 - 479s/epoch - 877ms/step
Epoch 97/100
547/547 - 481s - loss: 0.0304 - accuracy: 0.9976 - val_loss: 0.0260 - val_accuracy: 0.9956 - 481s/epoch - 878ms/step
Epoch 98/100
547/547 - 484s - loss: 0.0305 - accuracy: 0.9969 - val_loss: 0.0258 - val_accuracy: 0.9956 - 484s/epoch - 884ms/step
Epoch 99/100
547/547 - 485s - loss: 0.0298 - accuracy: 0.9970 - val_loss: 0.0256 - val_accuracy: 0.9958 - 485s/epoch - 887ms/step
Epoch 100/100
547/547 - 487s - loss: 0.0300 - accuracy: 0.9966 - val_loss: 0.0251 - val_accuracy: 0.9958 - 487s/epoch - 891ms/step
```

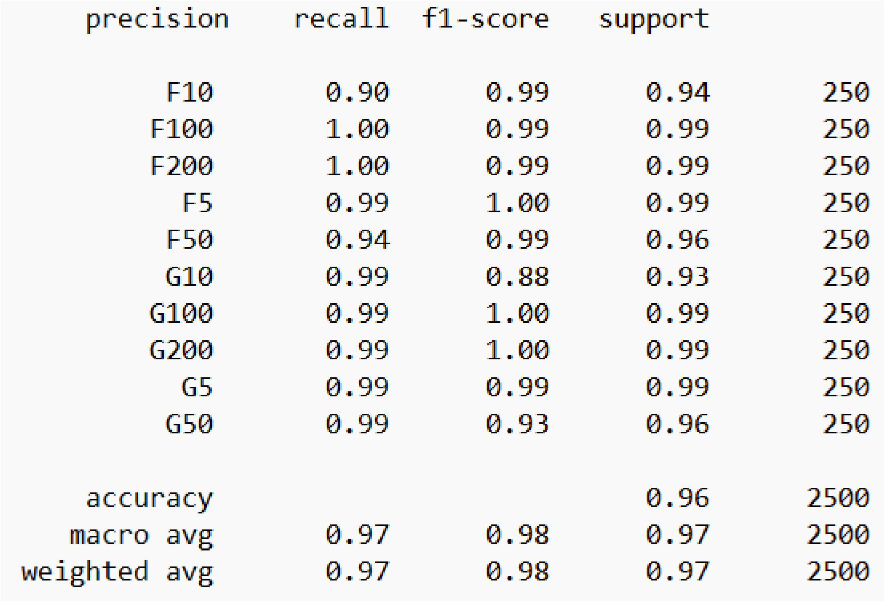
Figure 4- 10. Training performance of MobileNet at the last 10 epochs

The loss and accuracy metrics for the training and validation sets are shown in the output. The loss is a measure of how well the model fits the training data, and the accuracy is a measure of how well the model predicts the labels for the training data. The val_loss and val_accuracy metrics are the same for the validation set.

The loss and accuracy metrics for the training set are lower than the loss and accuracy metrics for the validation set. This suggests that the model is not overfitting the training data.

In the output you provided, the model was trained for 100 epochs. An epoch is a complete pass through the training data. The model was trained for 487 seconds, which is about 8 minutes. The model was able to achieve an accuracy of 99.66% on the training set and an accuracy of 99.58% on the validation set.

The model was able to achieve a good accuracy on both the training set and the validation set. This suggests that the model is not overfitting the training data and is able to generalize to new data.

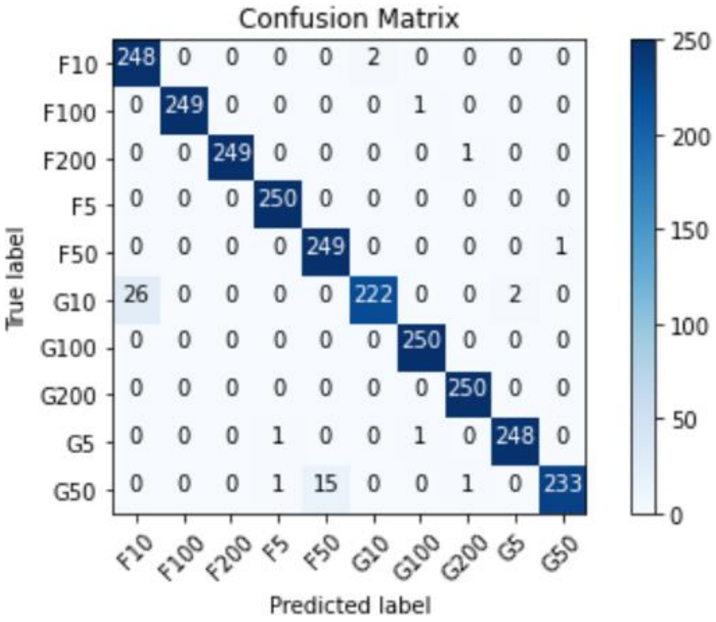


```
     precision    recall  f1-score   support

         F10       0.90      0.99      0.94       250
        F100       1.00      0.99      0.99       250
        F200       1.00      0.99      0.99       250
          F5       0.99      1.00      0.99       250
         F50       0.94      0.99      0.96       250
         G10       0.99      0.88      0.93       250
        G100       0.99      1.00      0.99       250
        G200       0.99      1.00      0.99       250
          G5       0.99      0.99      0.99       250
         G50       0.99      0.93      0.96       250

    accuracy                           0.96      2500
   macro avg       0.97      0.98      0.97      2500
weighted avg       0.97      0.98      0.97      2500
```

Figure 4- 11. Test Set Classification report using fine-tuned MobileNet

The F1 score for the F10 class is 0.94, the F100 class is 0.99, the F200 class is 0.99, the F5 class is 0.99, the F50 class is 0.96, the G10 class is 0.93, the G100 class is 0.99, the G200 class is 0.99, the G5 class is 0.99, and the G50 class is 0.96. The overall accuracy is 0.96. The macro average F1 score is 0.97 and the weighted average F1 score is 0.97. These results indicate that the model is performing well overall, with high accuracy and F1 scores across all classes.

The results indicate that the model is performing well. However, it is important to note that these results are based on a dataset. It is important to evaluate the model on a larger dataset to get a more accurate assessment of its performance. Detailed and clear evaluation result with their correct classification and misclassification rate in each class is presented below using a confusion matrix:



	F10	F100	F200	F5	F50	G10	G100	G200	G5	G50
F10	248	0	0	0	0	2	0	0	0	0
F100	0	249	0	0	0	0	1	0	0	0
F200	0	0	249	0	0	0	0	1	0	0
F5	0	0	0	250	0	0	0	0	0	0
F50	0	0	0	0	249	0	0	0	0	1
G10	26	0	0	0	0	222	0	0	2	0
G100	0	0	0	0	0	0	250	0	0	0
G200	0	0	0	0	0	0	0	250	0	0
G5	0	0	0	1	0	0	1	0	248	0
G50	0	0	0	1	15	0	0	1	0	233

Figure 4- 12. Confusion matrix for Test set classification with fine-tuned MobileNet

The presented confusion matrix illustrates a classification model's performance on a dataset. The analysis of the confusion matrix reveals the accuracy and errors made by the model for each class.

F10: The model correctly predicted 248 instances of class 0, indicating a high accuracy. However, there were 2 instances that were mistakenly classified as class 5, indicating some confusion between these two classes.

F100: All instances of class F100 (249) were correctly predicted, demonstrating excellent accuracy for this class.

F200: Similar to class F100, all instances of class 2 (249) were correctly predicted, indicating high accuracy.

F5: The model correctly predicted all instances of class F5 (250), indicating perfect accuracy for this class.

F50: Most instances of class F50 (249) were correctly predicted. However, there was 1 instance that was misclassified as class 9, suggesting a small error in the predictions for this class.

G10: The model performed well for class G10, correctly predicting 222 instances. However, there were 26 instances misclassified as class F10 and F200 instances misclassified as class G5, indicating some confusion between these classes.

G100: All instances of class G100 (250) were correctly predicted, demonstrating excellent accuracy for this class.

G200: Similar to class G100, all instances of class G200 (250) were correctly predicted, indicating high accuracy.

G5: The model performed well for class G5, correctly predicting 248 instances. However, there was 1 instance misclassified as class F5 and 1 instance misclassified as class G100, suggesting some confusion between these classes.

G50: The model correctly predicted 233 instances of class G50. However, there were 1 instance misclassified as class F5 and 15 instances misclassified as class F50, indicating some confusion between these classes.

Overall, the model demonstrates high accuracy for most classes.

4.5.4 Classification using Fine-tuned ResNet50

Lastly, we have also tested the Keras ResNet50 pre-trained model. Customizing the top 2 layers and fine-tuning the whole model to fit and converge the model for our 10-class classification problem. The model was set to be trained for a maximum of 100 epochs and achieved the following results:

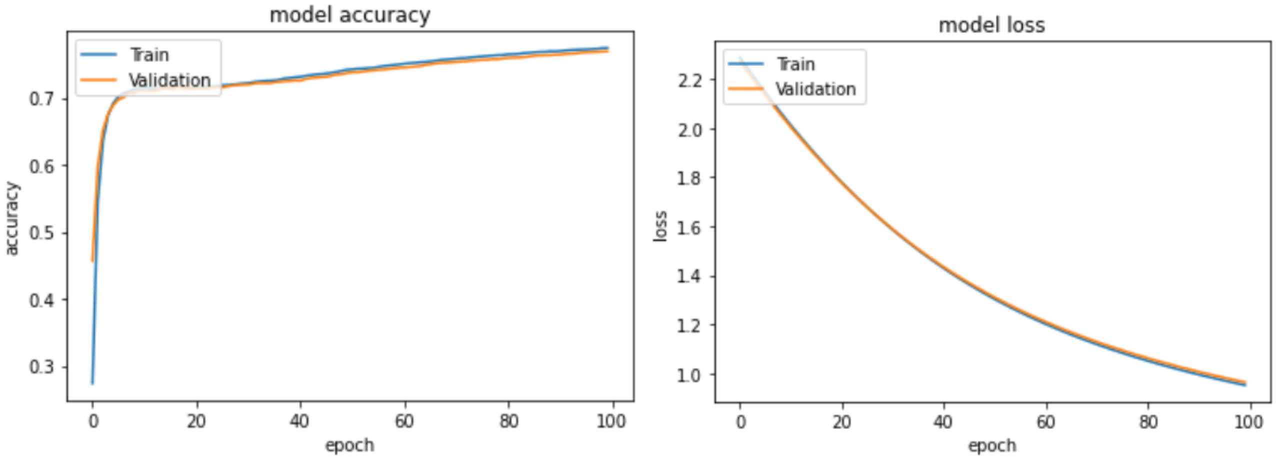
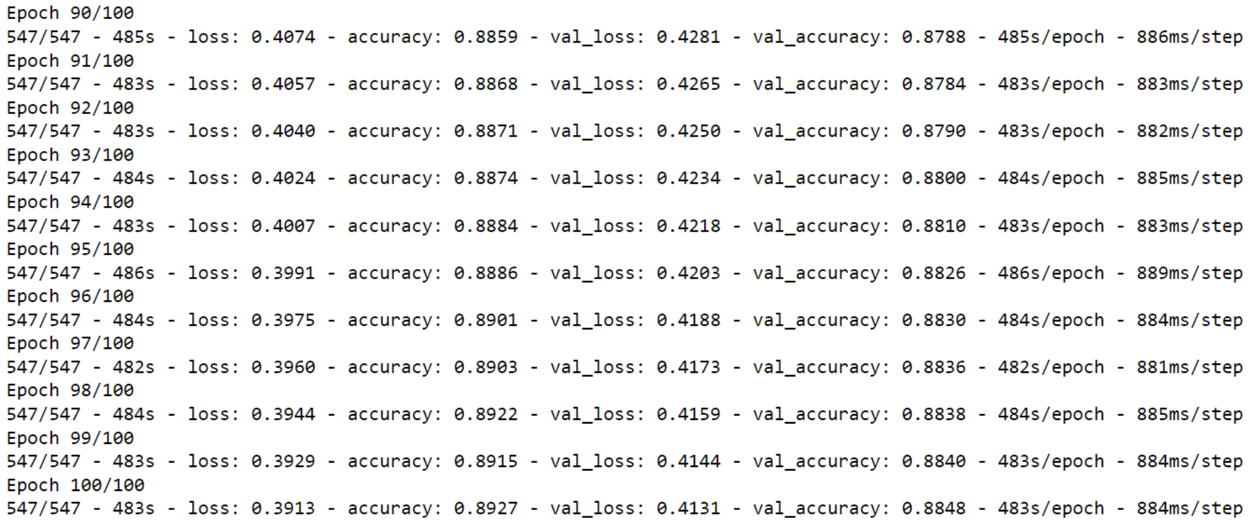


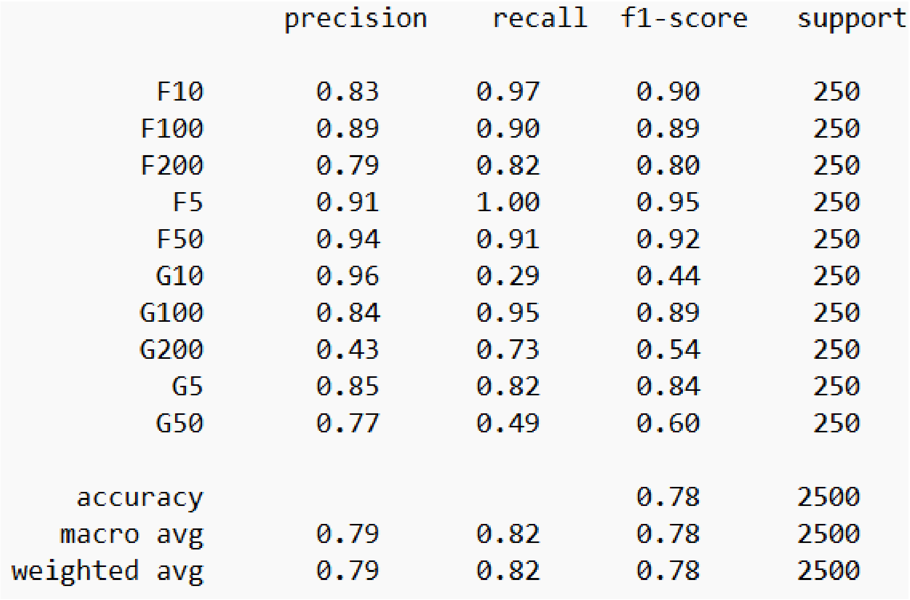
Figure 4- 13. ResNet50 training loss and accuracy curves

By observing the training loss and accuracy curve, it is evident that at epoch 100 with a learning rate of 0.0001 (which is significantly low), the final validation loss was reported as 0.41. Similarly, at epoch 100 with the same learning rate, the highest validation accuracy of 88.48% and training accuracy of 89.27% were achieved.



```
Epoch 90/100
547/547 - 485s - loss: 0.4074 - accuracy: 0.8859 - val_loss: 0.4281 - val_accuracy: 0.8788 - 485s/epoch - 886ms/step
Epoch 91/100
547/547 - 483s - loss: 0.4057 - accuracy: 0.8868 - val_loss: 0.4265 - val_accuracy: 0.8784 - 483s/epoch - 883ms/step
Epoch 92/100
547/547 - 483s - loss: 0.4040 - accuracy: 0.8871 - val_loss: 0.4250 - val_accuracy: 0.8790 - 483s/epoch - 882ms/step
Epoch 93/100
547/547 - 484s - loss: 0.4024 - accuracy: 0.8874 - val_loss: 0.4234 - val_accuracy: 0.8800 - 484s/epoch - 885ms/step
Epoch 94/100
547/547 - 483s - loss: 0.4007 - accuracy: 0.8884 - val_loss: 0.4218 - val_accuracy: 0.8810 - 483s/epoch - 883ms/step
Epoch 95/100
547/547 - 486s - loss: 0.3991 - accuracy: 0.8886 - val_loss: 0.4203 - val_accuracy: 0.8826 - 486s/epoch - 889ms/step
Epoch 96/100
547/547 - 484s - loss: 0.3975 - accuracy: 0.8901 - val_loss: 0.4188 - val_accuracy: 0.8830 - 484s/epoch - 884ms/step
Epoch 97/100
547/547 - 482s - loss: 0.3960 - accuracy: 0.8903 - val_loss: 0.4173 - val_accuracy: 0.8836 - 482s/epoch - 881ms/step
Epoch 98/100
547/547 - 484s - loss: 0.3944 - accuracy: 0.8922 - val_loss: 0.4159 - val_accuracy: 0.8838 - 484s/epoch - 885ms/step
Epoch 99/100
547/547 - 483s - loss: 0.3929 - accuracy: 0.8915 - val_loss: 0.4144 - val_accuracy: 0.8840 - 483s/epoch - 884ms/step
Epoch 100/100
547/547 - 483s - loss: 0.3913 - accuracy: 0.8927 - val_loss: 0.4131 - val_accuracy: 0.8848 - 483s/epoch - 884ms/step
```

Figure 4- 14. Training performance of ResNet50 at the last 10 epochs



```
              precision    recall  f1-score   support

         F10       0.83      0.97      0.90       250
        F100       0.89      0.90      0.89       250
        F200       0.79      0.82      0.80       250
          F5       0.91      1.00      0.95       250
         F50       0.94      0.91      0.92       250
         G10       0.96      0.29      0.44       250
        G100       0.84      0.95      0.89       250
        G200       0.43      0.73      0.54       250
          G5       0.85      0.82      0.84       250
         G50       0.77      0.49      0.60       250

    accuracy                           0.78      2500
   macro avg       0.79      0.82      0.78      2500
weighted avg       0.79      0.82      0.78      2500
```

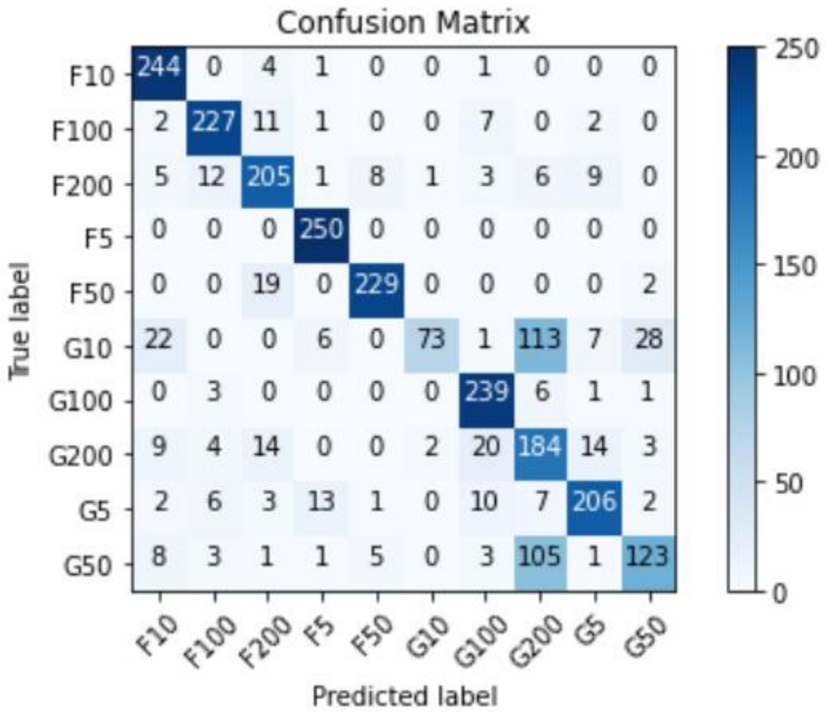
Figure 4- 15. Test Set Classification report for fine-tuned ResNet50

This model achieved excellent results in predicting class F10 with a precision of 83%, recall of 97%, and an F1-score of 90%. These metrics indicate that the model has a high accuracy in identifying instances of class F10. The model performed well in predicting class F100 with a precision of 89% and a recall of 90%. This indicates that the model has a good ability to identify instances of class F100. The model achieved a precision of 79% and a recall of 82% for class F200. While the precision and recall values are relatively lower compared to other classes, they still indicate a reasonable performance in identifying instances of class F200. The model demonstrated high accuracy in predicting class F5, with a precision of 91%, recall of 100%, and an F1-score of 95%. These metrics suggest that the model is highly capable of identifying instances of class F5. The model performed well in predicting class F50, with a precision of 94%, recall of 91%, and an F1-score of 92%. These metrics indicate a good ability to identify instances of class F50. The model achieved a high precision of 96% for class G10, indicating that most predicted instances are correct. However, the recall is relatively low at 29%, indicating that the model struggled to identify a significant portion of actual instances of class G10. The model demonstrated good performance in predicting class G100, with a precision of 84% and a recall of 95%. These metrics suggest a relatively high accuracy in identifying instances of class G100. The model achieved a precision of 43% and a recall of 73% for class G200. These metrics indicate that the model had difficulty in accurately identifying instances of class G200. The model performed reasonably well in predicting class G5, with a precision of 85%, recall of 82%, and an F1-score of 84%. These metrics indicate a decent ability to identify instances of class G5. The model had a precision of 77% and a recall of 49% for class G50. These metrics suggest that the model had challenges in accurately identifying instances of class G50.

The model achieved an average test accuracy of 78%, which is close to the validation accuracy. This indicates that the model's performance on the test set is consistent with its performance on the validation set.

Detailed and clear evaluation result with their correct classification and misclassification rate in each class is presented below using a confusion matrix:

The confusion matrix clearly shows, the highest error distribution was seen in class G10 classification as some images were G200 as G50 and F10, and the other highest error distribution was seen in class G50 classification as some images were G200 whereas the lowest error distribution is for class F5 classification as 0 image were misclassified. From this, we can understand that the model is a confused to differentiate some features between class G200, G10 and G50, even though an average test accuracy of 89.2% was achieved.



True label \ Predicted label	F10	F100	F200	F5	F50	G10	G100	G200	G5	G50
F10	244	0	4	1	0	0	1	0	0	0
F100	2	227	11	1	0	0	7	0	2	0
F200	5	12	205	1	8	1	3	6	9	0
F5	0	0	0	250	0	0	0	0	0	0
F50	0	0	19	0	229	0	0	0	0	2
G10	22	0	0	6	0	73	1	113	7	28
G100	0	3	0	0	0	0	239	6	1	1
G200	9	4	14	0	0	2	20	184	14	3
G5	2	6	3	13	1	0	10	7	206	2
G50	8	3	1	1	5	0	3	105	1	123

Figure 4- 16. Confusion matrix for Test classification with fine-tuned ResNet50

From these four fine-tuning experiments used, the better testing accuracy 99.6% was achieved using fine-tuned MobileNet, whereas InceptionV3 performed weak with 41.2% testing accuracy. From all experiments we have seen, the accuracy of deep feature engineering on different pre-trained networks MobileNet, VGG16, and ResNet50 models show a better performance with less size, number of parameters, and training time than the other pre-trained models.

Table 4- 2. Comparison of pre-trained Models Final result summary

Pre-trained Model	Optimizer	Best Epoch	Accuracy		Loss	
			Training	Validation	Training	Validation
InceptionV3	Adam	100	0.41	1.63	0.40	1.63
	SGD	100	0.41	1.63	0.40	1.63
VGG16	Adam	60	0.98	0.75	0.05	6.50
	SGD	100	0.99	0.98	0.01	0.09
MobileNet	Adam	100	0.97	0.95	0.08	0.19
	SGD	100	0.99	0.99	0.03	0.02
ResNet50	Adam	100	0.77	0.77	0.95	0.96
	SGD	100	0.89	0.88	0.39	0.88

The table 4-2. Provides an overview of the performance of different pre-trained models with their respective parameters and optimizers. When tested, it was observed that the Adam optimizer performed relatively weaker compared to SGD. Although the results were comparable for all fine-tuned models, MobileNet showed better performance when paired with the SGD optimizer.

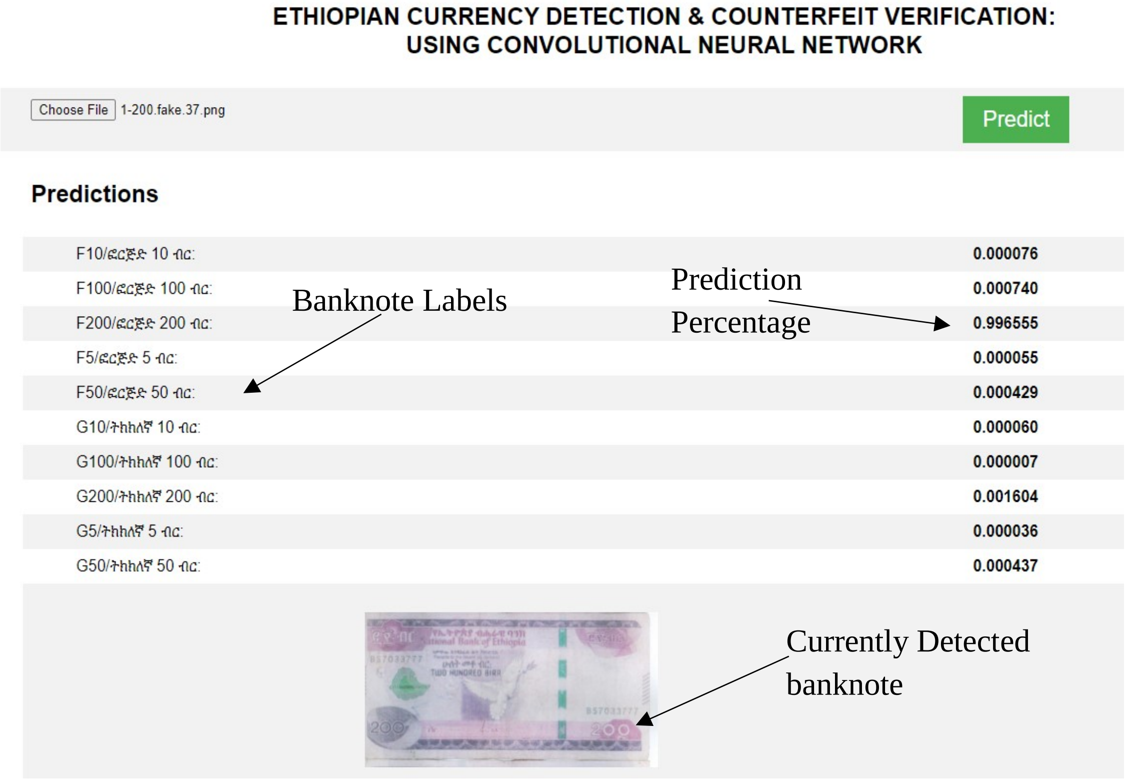
4.6 Implementation of Web-Based and Android Application

The MobileNet model, a deep learning architecture optimized with SGD (Stochastic Gradient Descent), has been purposefully selected for its exceptional capabilities in detecting Ethiopian banknotes. This model, known for its efficiency and accuracy in image recognition tasks, has been specifically implemented to cater to two distinct platforms: a web-based application Figure 4-16 and Android mobile applications Figure 4-17.

By utilizing the MobileNet model with SGD optimization, the web-based application and Android applications can effectively analyze and process images of Ethiopian banknotes. These applications leverage the power of deep learning algorithms to accurately identify and classify various security elements present on both the front and back of the banknotes.

The integration of the MobileNet model in the web-based application and Android applications empowers users to easily and conveniently verify the authenticity of Ethiopian banknotes. Whether accessed through a web browser or installed as an application on an Android mobile device, these tools offer a user-friendly interface for capturing images of banknotes and swiftly determining their genuineness.

The MobileNet model, combined with SGD optimization, ensures that the banknote detection process is not only efficient but also highly reliable, allowing for the differentiation between genuine banknotes and counterfeit ones. This utilization of advanced deep learning techniques contributes to enhancing the overall security and confidence in handling Ethiopian currency.



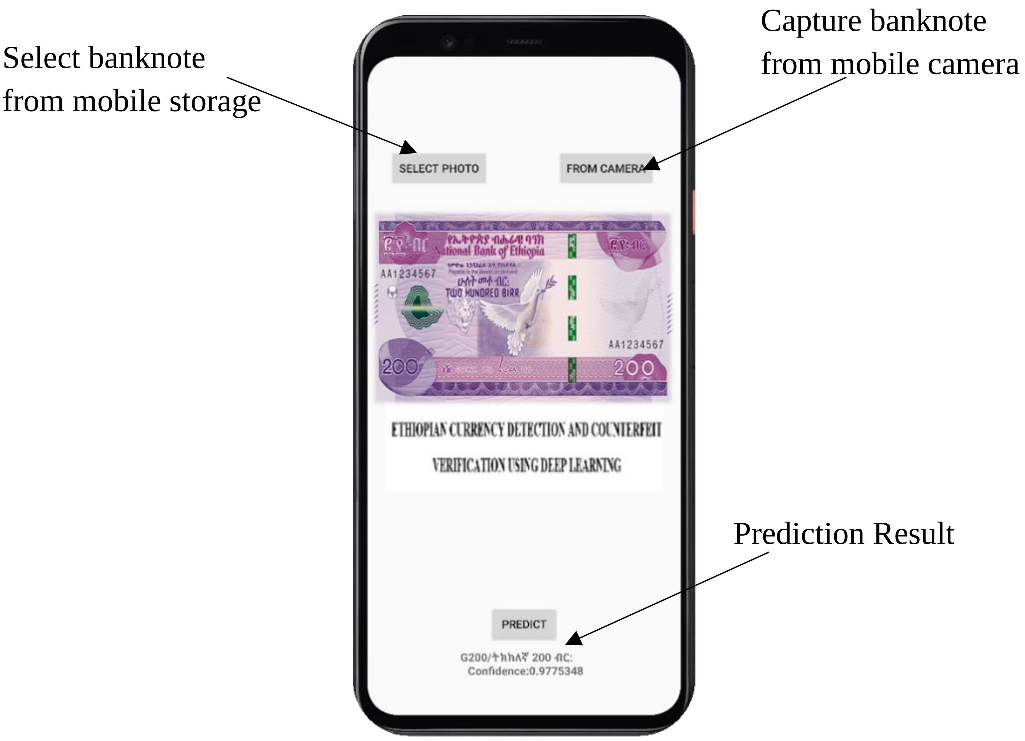
ETHIOPIAN CURRENCY DETECTION & COUNTERFEIT VERIFICATION:
USING CONVOLUTIONAL NEURAL NETWORK

Choose File 1-200.fake.37.png

Predictions

F10/ፎርጅድ 10 ብር:	0.000076
F100/ፎርጅድ 100 ብር:	0.000740
F200/ፎርጅድ 200 ብር:	0.996555
F5/ፎርጅድ 5 ብር:	0.000055
F50/ፎርጅድ 50 ብር:	0.000429
G10/ትክክለኛ 10 ብር:	0.000060
G100/ትክክለኛ 100 ብር:	0.000007
G200/ትክክለኛ 200 ብር:	0.001604
G5/ትክክለኛ 5 ብር:	0.000036
G50/ትክክለኛ 50 ብር:	0.000437

Figure 4- 17. Web-based application



ETHIOPIAN CURRENCY DETECTION AND COUNTERFEIT VERIFICATION USING DEEP LEARNING

G200/ትክክለኛ 200 ብር:
Confidence:0.9775348

Figure 4- 18. Android mobile application

4.7 Discussion and Summary

In this chapter, we provide a comprehensive explanation of the experimental results obtained from the proposed model. The model was trained and tested, and its performance was evaluated using various performance metrics. Additionally, we compared the performance of the proposed model with that of pre-trained models, considering the same metrics. Overall, the results showed that the fine-tuned MobileNet model outperformed the others, while InceptionV3 exhibited weaker testing accuracy. Based on the experiments conducted, it can be concluded that deep feature engineering using the MobileNet model yielded superior results compared to other pre-trained models, with advantages such as smaller size, fewer parameters, and shorter training time.

CHAPTER FIVE

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this research study, a novel approach utilizing deep learning technology has been developed to differentiate and classify genuine banknotes from counterfeits. The main objective was to create a prototype Ethiopian banknote recognition system capable of accurately categorizing different denominations of Ethiopian banknotes. To achieve this, a dataset of approximately 30,000 genuine and counterfeit images of Ethiopian paper currency was collected and organized for training, testing, and validation based on their denominations.

The evaluation of the suggested method involved training the created framework using four distinct convolutional neural network (CNN) designs: InceptionV3, VGG16, MobileNet, and ResNet50. To optimize the training process for each architecture, two techniques were used: Adam and SGD.

The aim of this research was to develop an effective and efficient system for Ethiopian banknote detection and counterfeit verification using convolutional neural network that can accurately identify Ethiopian bank note denominations as well as detect counterfeit currency.

Among all the tested CNN architectures and optimization techniques, the MobileNet architecture combined with the SGD optimization technique and a batch size of 32 demonstrated the highest training accuracy of 99.66% and overall accuracy 97%. This reliable and robust system has various potential applications, including its use in ATMs for cash deposits and withdrawals, financial transactions, banknote counters, smart transportation systems, and automatic vending machines that accept banknotes as payment.

The results of this research are anticipated to have important implications for the advancement of efficient and fast banknote recognition systems utilizing banknote images. Moreover, these findings can contribute to the progress of more precise point-of-care detection tools that are specifically tailored for recognizing and classifying Ethiopian banknotes.

5.2 Future work

The developed model of the study is Ethiopian banknote detection and counterfeit verification using convolutional neural network that can accurately identify Ethiopian bank note denominations as well as detect counterfeit currency, and the study achieves its objectives in a better way since the main objective of the study was developing a Ethiopian banknote detection and counterfeit verification model using CNN and the study achieved its objective. For future study related to this topic we recommended:

- Researchers can explore more advanced machine learning techniques such as deep learning, reinforcement learning, and generative adversarial networks (GANs) to enhance the accuracy and robustness of counterfeit detection systems.
- Integrating multiple modalities, such as combining visual information from images with other sensor data like infrared or ultraviolet scans, can improve the detection capabilities and make counterfeit detection systems more reliable.
- Developing real-time counterfeit detection systems that can quickly and accurately identify counterfeit banknotes during transactions or at points of sale can help prevent financial losses and protect businesses and consumers.
- Increasing the size and diversity of the training datasets used for developing counterfeit detection models can help improve their generalization and robustness. Collecting more banknote samples from different sources and variations can lead to better detection performance.
- With the rise of digital payment systems and online transactions, developing methods to detect counterfeit digital currencies or identify fraudulent transactions can be an important area of research.

References

[1] Kang, Heung-Sik, Chang-Ki Min, Hoon Heo, Changbum Kim, Haeryong Yang, Gyujin Kim and Inhyuk Nam, "Hard X-ray free-electron laser with femtosecond-scale timing jitter," *Nature Photonics,* vol. 11, no. 11, pp. 708-713, 2017.

[2] F. Jegnaw and A. Yaregal, "Automatic recognition and counterfeit detection of Ethiopian paper currency," *International Journal of Image, Graphics and Signal Processing,* vol. 8, no. 2, p. 28, 2016.

[3] H. Hassanpour, A. Yaseri and G. Ardeshiri, "Feature extraction for paper currency recognition," *2007 9th International Symposium on Signal Processing and Its Applications, Sharjah, United Arab Emirates,* vol. 10, no. 1109/ISSPA.2007.4555366, pp. 1-4, 2007.

[4] Hlaing, N. Khin and A. Gopalakrishnan, "Myanmar paper currency recognition using GLCM and k-NN," *2016 Second Asian Conference on Defence Technology (ACDT), Chiang Mai, Thailand,* vol. 10, no. 1109/ACDT.2016.7437645, pp. 67-72, 2016.

[5] Ali, E. Syed, G. Mriganka and M. Subra, "Challenges in Indian currency denomination recognition & authentication," *International Journal of Research in Engineering and Technology,* vol. 3, no. 11, pp. 477-483, 2014.

[6] Daliyah Aljutaili, Redna Almutlaq, Suha Alharbi and Dina Ibrahim, "A Speeded up Robust Scale-Invariant Feature A Speeded up Robust Scale-Invariant Feature," *International Journal of Computer and Information Engineering*, 2018.

[7] Chaitali Kadam and Prof. B. Borse, "A Comparative Study of Image Denoising Techniques for Medical Image," *International Research Journal of Engineering and Technology,* p. 369, 2017 .

[8] Ramandeep Kaur, Seema Baghla and Sunil Kumar, "A Review on skiew detection and correction in Multiscript text document script. International Journal of Advances in Science Engineering and Technology," *International Journal of Advances in Science, Engineering and Technology(IJASEAT),* vol. 3, no. 3, pp. 145-148, 2015.

[9] Gajendra Singh, Ravindra Kumar, Deepika Khare and Sumita Verma, "Analysis of Image Segmentation Algorithms Using MATLAB," *International Journal of Engineering Innovation & Research,* vol. 1, no. 1, pp. 51-55, 2012.

[10] Rajkumar Goel, Vineet Kumar, Saurabh Srivastava and K. Sinha, "A Review of Feature Extraction Techniques for Image analysis," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 6, no. 2, pp. 153-155, 2017.

[11] Varshni, D., Thakral, K., Agarwal, L., Nijhawan and R, "Pneumonia detection using CNN based feature extraction," *IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT),* pp. 1-7, 2019.

[12] Jian Kang Wu, Mohan S. Kankanhalli, Joo-Hwee Lim and Dezhong Hong, "Perspectives on Content-Based Multimedia Systems," *Springer, Boston, MA,* vol. 1, no. 9, 2000.

[13] Hanish Aggarwal and Padam Kumar, "Indian Currency Note Denomination Recognition in Color Images," *IEEE,* pp. 1-5, 2016.

[14] Pragati Pawar and Shrikant Kale, "Recognition of Indian Currency Note Based on HSV Parameters," *International Journal of Science and Research,* vol. 3, no. 6, pp. 132-137, 2014.

[15] Wang, S, Liew and Alan Wee-Chung, "Information-based Color Feature representation for Image classification," *Proceedings of the 2007 IEEE International conference on Image Processing,* p. 353, 2007.

[16] Mitisha Narottambhai and Purvi Tandel, "A Survey on Feature Extraction Techniques for Shape absed object recognition technique," *International Journal of Computer Applications,* vol. 6, p. 137, 2016.

[17] Castelli and L. D. Bergman, "Image Databases: Search and Retrieval of Digital Imagery," *New york: Wiley,* 2012.

[18] Rikiya Yamashita and M. N. , "Convolutional neural networks: an overview and application in radiology," *National Center for Biotechnology Information, U.S National Library of Medicine,* vol. 9, no. 4, pp. 611-629, 2018.

[19] P. Ravindra, "Deep Learning," *Deep Learning,* 2018.

[20] Chen Wang and Yang Xi, "Convolutional Neural Network for Image Classification," *Baltimore: Johns Hopkins University, USA,* 2018.

[21] Pawar, D. Pragati and B. K. Shrikant, "Recognition of Indian currency note based on HSV parameters," *International Journal of Science and Research,* vol. 3, no. 6, pp. 132-137, 2014.

[22] Y. Liang, L. Liu, , Y. Xu, Y. Xiang and B. Zou, "Multi-task gloh feature selection for human age estimation," *18th IEEE International Conference on Image Processing, Brussels, Belgium,* p. 565–568, 2011.

[23] E. Choi, J. Lee and J. Yoon, "Feature extraction for banknote classification using wavelet transform," *18th International Conference on Pattern Recognition, Hong Kong, China,* p. 934–937, 2006.

[24] D. A. K. S. Gunaratna, K. Nihal and P. H.L. , "ANN Based Currency Recognition System using Compressed Gray Scale and Application for Sri Lankan Currency Notes - SLCRec," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering,* 2008.

[25] M. S. Uddin, P. P. Das and M. S. A. Roney, "Image-based approach for the detection of counterfeit banknote of Bangladesh," *5th International Conference on Informatics, Electronics and Vision (ICIEV),* p. 1067–1072, 2016.

[26] Young Ho Park, Seung Yong Kwon and Tuyen Danh Pham , "A High Performance Banknote Recognition System Based on a One-Dimensional Visible Light Line Sensor," *Imaging: Sensors and Technologies,* vol. 15(6), pp. 14093-14115, 2015.

[27] S. Gai, G. Yang and M. Wan, "Employing quaternion wavelet transform for banknote classification," *Neurocomputing,* vol. 118, p. 171–178, 2013.

[28] J. He, H. Zhang, O. Jin, Z. Hu and J. Zhang, "A novel recognition method for Nigeria paper currency based on support vector machine & genetic algorithms," *International Symposium on Signal Processing, Biomedical Engineering and Informatics,* vol. 6, 2013.

[29] R. V. K. Reddy, B. S. Rao and K. P. Raju, "Handwritten Hindi digits recognition using convolutional neural network with RMSprop optimization," *018 Second International Conference on Intelligent Computing and Control Systems (ICICCS),* p. 45–51, 2018.

[30] C. L. Huang, Y. C. Shih and C. M. Lai, "Optimization of a convolutional neural network using a hybrid algorithm," *International Joint Conference on Neural Networks,* vol. 1, p. 14–19, 2019.

[31] AM Taqi, A Awad, F Al-Azzo and M Milanova, "The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance," *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 140-145, 2018.

[32] Aseffa, T. Dereje, K. Harish and M. Satyasis, "Ethiopian banknote recognition using convolutional neural network and its prototype development using embedded platform," *Journal of Sensors*, pp. 1-18, 2022.

[33] M. Million and S. Asfaw , "Ethiopian Paper Currency Recognition System: An Optimal Feature Extraction," *IEEE-SEM* , vol. vol. 7, p. no. 8, 2019.

[34] M. L. Meharu and H. S. Worku, "Real-time Ethiopian currency recognition for visually disabled peoples using convolutional neural network," pp. 47-52, 2020.

[35] . J. Raymond, "The importance of Intaglio in the authentication of banknotes by the general public," *Prepared by Secure Perception Research for International Banknote Designers Association*, 2017.

[36] E. A. Tessfaw, B. Ramani, and T. K. Bahiru, "Ethiopian banknote recognition and fake detection using support vector machine," *In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, vol. 20, pp. 1354-1359, 2018.

[37] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe and Jonathon Shlens, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE conference on computer vision and pattern* , p. 2818–2826, 2016.

[38] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Proceedings of the IEEE conference on computer vision and pattern* , vol. V6, 2015.

[39] Xinzheng Xu, Meng Du, Huanxiu Guo and Jianying Chang, "Lightweight FaceNet Based on MobileNet," *International Journal of Intelligence Science*, vol. 11, 2020.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE conference on computer vision and pattern* , 2015 .

[41] P. Ciblat and P. Loubaton, "Second order blind equalization: the band limited case," *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181),* 2002.

[42] A. Alene, "Designing Ethiopian banknote classification and counterfeit verification system: an optimal feature extraction and classification techniques," *Doctoral dissertation,* pp. 13-18, 2019.

[43] M. Yewondwossen, "History and fresh face of Ethiopia Currency," Capitalethiopia, 21 Sept 2020. [Online]. Available: https://www.capitalethiopia.com/news-news/the-history-and-fresh-face-of-ethiopian-currency/. [Accessed 16 Sept 2022].

[44] Shin, Hoo-Chang and et al, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Transactions on Medical Imaging,* vol. 35, no. 5, pp. 1285-1298, 2016.

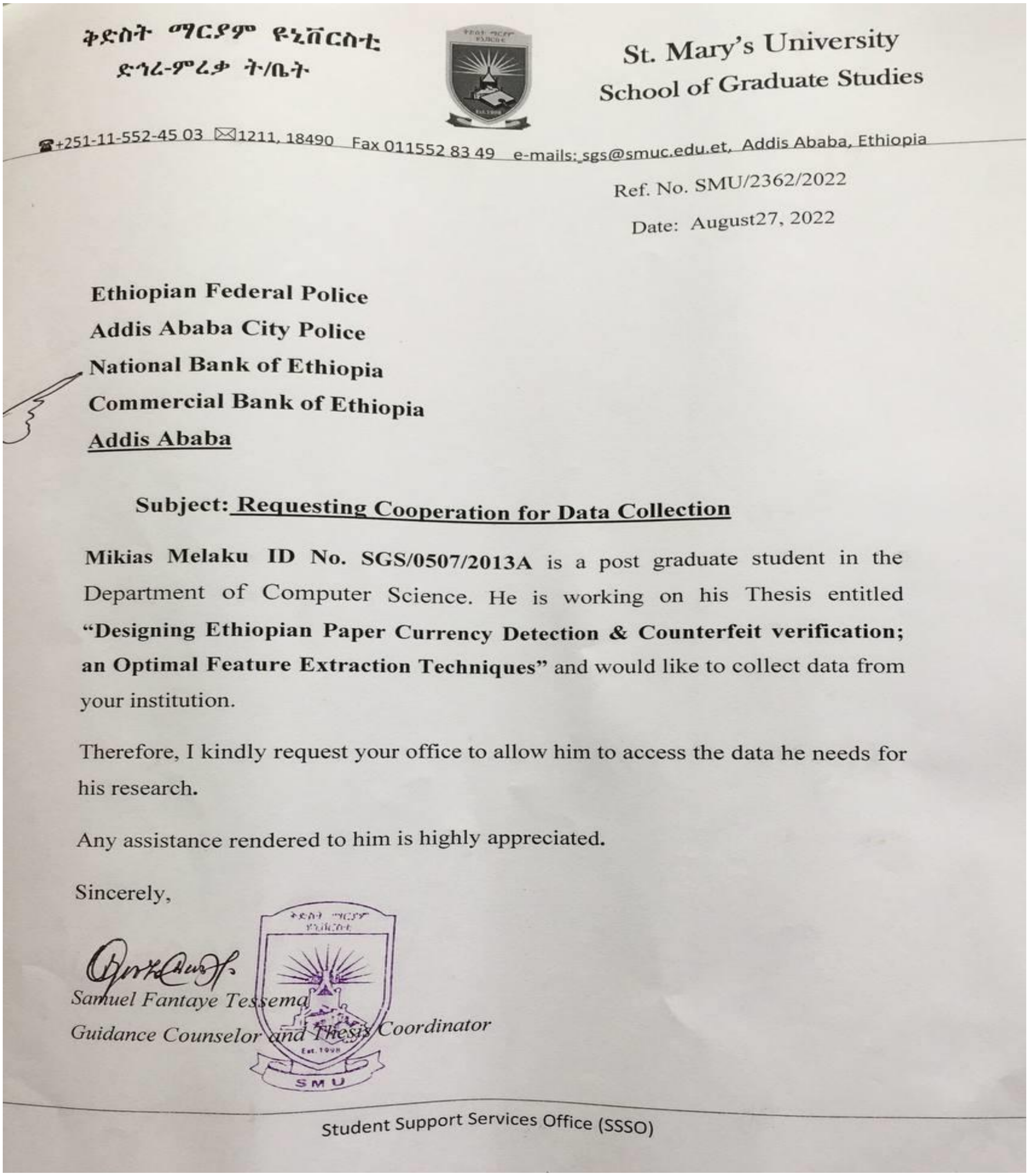
[45] Y. Muluken , "History and fresh face of Ethiopia Currency," Capitalethiopia, 21 09 2020. [Online]. Available: https://www.capitalethiopia.com/2020/09/21/the-history-and-fresh-face-of-ethiopian-currency/. [Accessed 03 02 2023].

[46] D. Jifeng , L. Yi , H. Kaiming and S. Jian , "Object Detection via Region-based Fully Convolutional Networks," *Cornel University* , vol. V2, 2016.

[47] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, no. 10, pp. 1345 - 1359, 2009.

Appendix

Support Letters for Ethiopian Federal Police, Addis Ababa City Police, National Bank of Ethiopia, Commercial Bank of Ethiopia



ቅድስት ማርያም ዩኒቨርስቲ
ድኅረ-ምረቃ ት/ቤት

St. Mary's University
School of Graduate Studies

☎+251-11-552-45 03 ✉1211, 18490 Fax 011552 83 49 e-mails: sgs@smuc.edu.et, Addis Ababa, Ethiopia

Ref. No. SMU/2362/2022

Date: August27, 2022

Ethiopian Federal Police
Addis Ababa City Police
National Bank of Ethiopia
Commercial Bank of Ethiopia
Addis Ababa

Subject: Requesting Cooperation for Data Collection

Mikias Melaku ID No. SGS/0507/2013A is a post graduate student in the Department of Computer Science. He is working on his Thesis entitled **"Designing Ethiopian Paper Currency Detection & Counterfeit verification; an Optimal Feature Extraction Techniques"** and would like to collect data from your institution.

Therefore, I kindly request your office to allow him to access the data he needs for his research.

Any assistance rendered to him is highly appreciated.

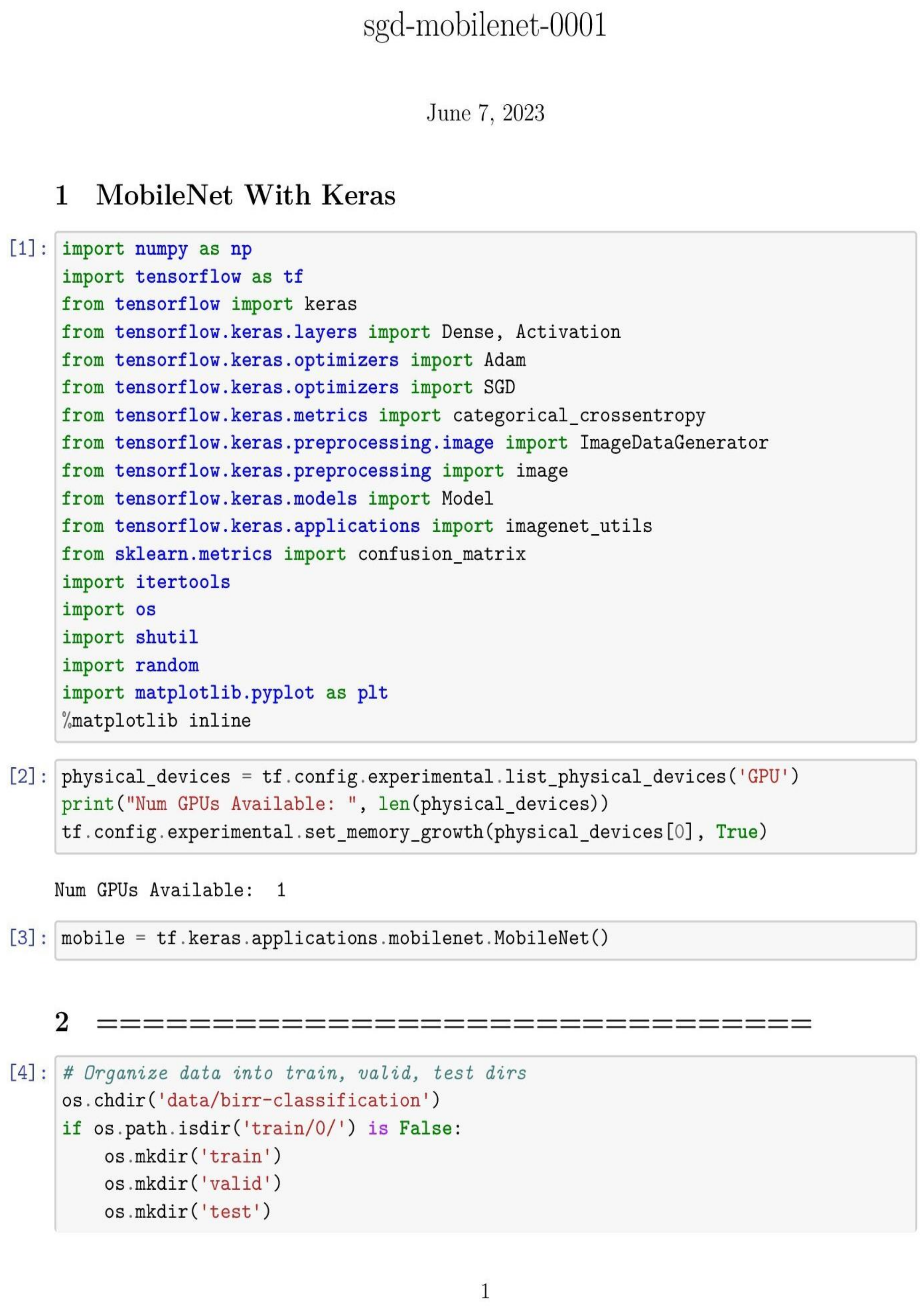
Sincerely,

Samuel Fantaye Tessema
Guidance Counselor and Thesis Coordinator

Est. 1998
SMU

Student Support Services Office (SSSO)

MobileNet Fine-Tuning with SGD Optimizer



sgd-mobilenet-0001

June 7, 2023

1 MobileNet With Keras

[1]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.applications import imagenet_utils
from sklearn.metrics import confusion_matrix
import itertools
import os
import shutil
import random
import matplotlib.pyplot as plt
%matplotlib inline
```

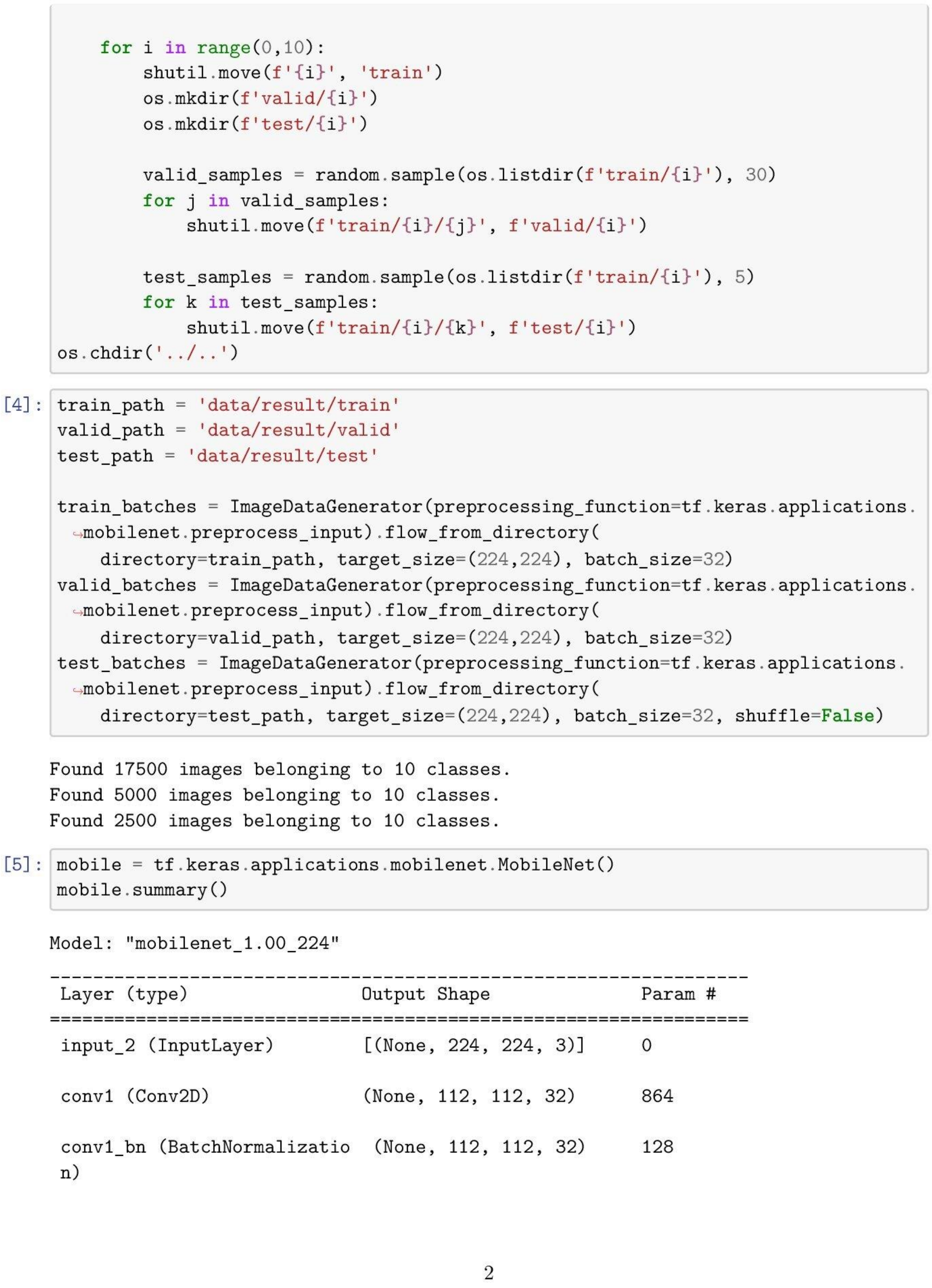
[2]:
```
physical_devices = tf.config.experimental.list_physical_devices('GPU')
print("Num GPUs Available: ", len(physical_devices))
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

```
Num GPUs Available:  1
```

[3]:
```
mobile = tf.keras.applications.mobilenet.MobileNet()
```

2 ==================================

[4]:
```
# Organize data into train, valid, test dirs
os.chdir('data/birr-classification')
if os.path.isdir('train/0/') is False:
    os.mkdir('train')
    os.mkdir('valid')
    os.mkdir('test')
```



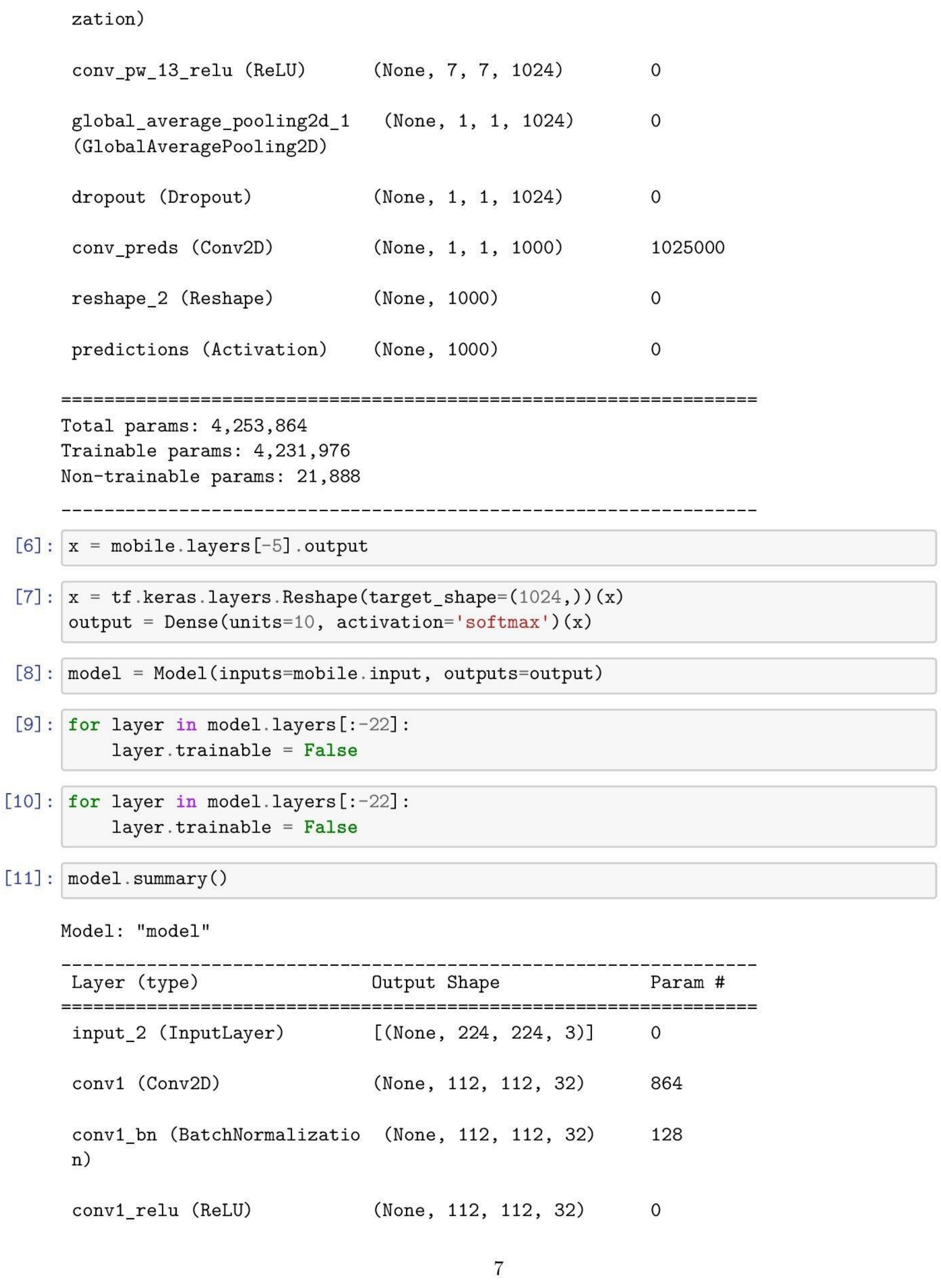
```
    for i in range(0,10):
        shutil.move(f'{i}', 'train')
        os.mkdir(f'valid/{i}')
        os.mkdir(f'test/{i}')

        valid_samples = random.sample(os.listdir(f'train/{i}'), 30)
        for j in valid_samples:
            shutil.move(f'train/{i}/{j}', f'valid/{i}')

        test_samples = random.sample(os.listdir(f'train/{i}'), 5)
        for k in test_samples:
            shutil.move(f'train/{i}/{k}', f'test/{i}')
os.chdir('../..')
```

```
[4]: train_path = 'data/result/train'
     valid_path = 'data/result/valid'
     test_path = 'data/result/test'

     train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
      ↪mobilenet.preprocess_input).flow_from_directory(
         directory=train_path, target_size=(224,224), batch_size=32)
     valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
      ↪mobilenet.preprocess_input).flow_from_directory(
         directory=valid_path, target_size=(224,224), batch_size=32)
     test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
      ↪mobilenet.preprocess_input).flow_from_directory(
         directory=test_path, target_size=(224,224), batch_size=32, shuffle=False)
```

```
Found 17500 images belonging to 10 classes.
Found 5000 images belonging to 10 classes.
Found 2500 images belonging to 10 classes.
```

```
[5]: mobile = tf.keras.applications.mobilenet.MobileNet()
     mobile.summary()
```

```
Model: "mobilenet_1.00_224"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizatio  (None, 112, 112, 32)     128
 n)
```



```
 zation)

 conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 global_average_pooling2d_1   (None, 1, 1, 1024)       0
 (GlobalAveragePooling2D)

 dropout (Dropout)           (None, 1, 1, 1024)        0

 conv_preds (Conv2D)         (None, 1, 1, 1000)        1025000

 reshape_2 (Reshape)         (None, 1000)              0

 predictions (Activation)    (None, 1000)              0

=================================================================
Total params: 4,253,864
Trainable params: 4,231,976
Non-trainable params: 21,888
_________________________________________________________________
```

```
[6]: x = mobile.layers[-5].output
```

```
[7]: x = tf.keras.layers.Reshape(target_shape=(1024,))(x)
     output = Dense(units=10, activation='softmax')(x)
```

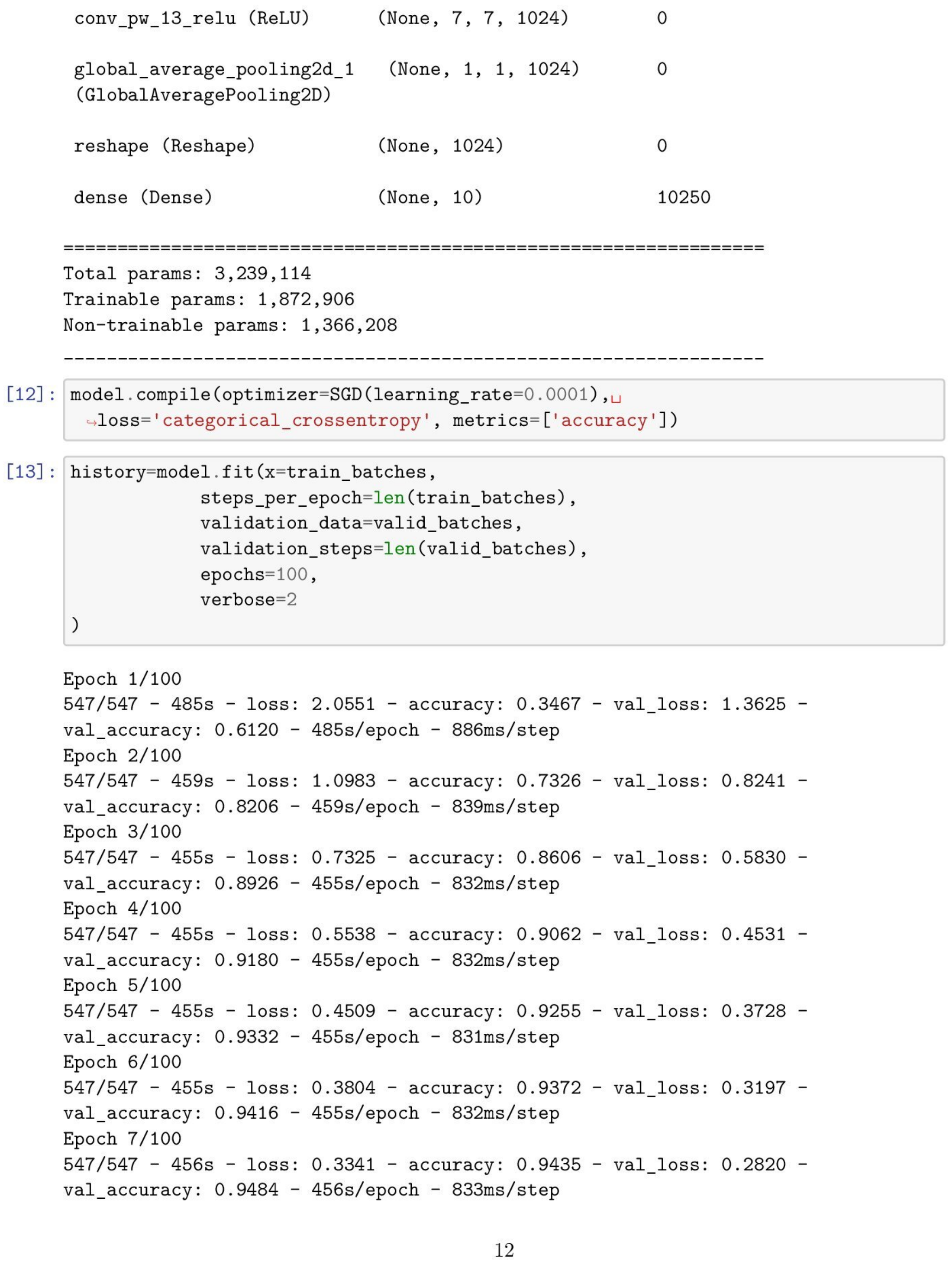
```
[8]: model = Model(inputs=mobile.input, outputs=output)
```

```
[9]: for layer in model.layers[:-22]:
         layer.trainable = False
```

```
[10]: for layer in model.layers[:-22]:
          layer.trainable = False
```

```
[11]: model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizatio  (None, 112, 112, 32)     128
 n)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0
```

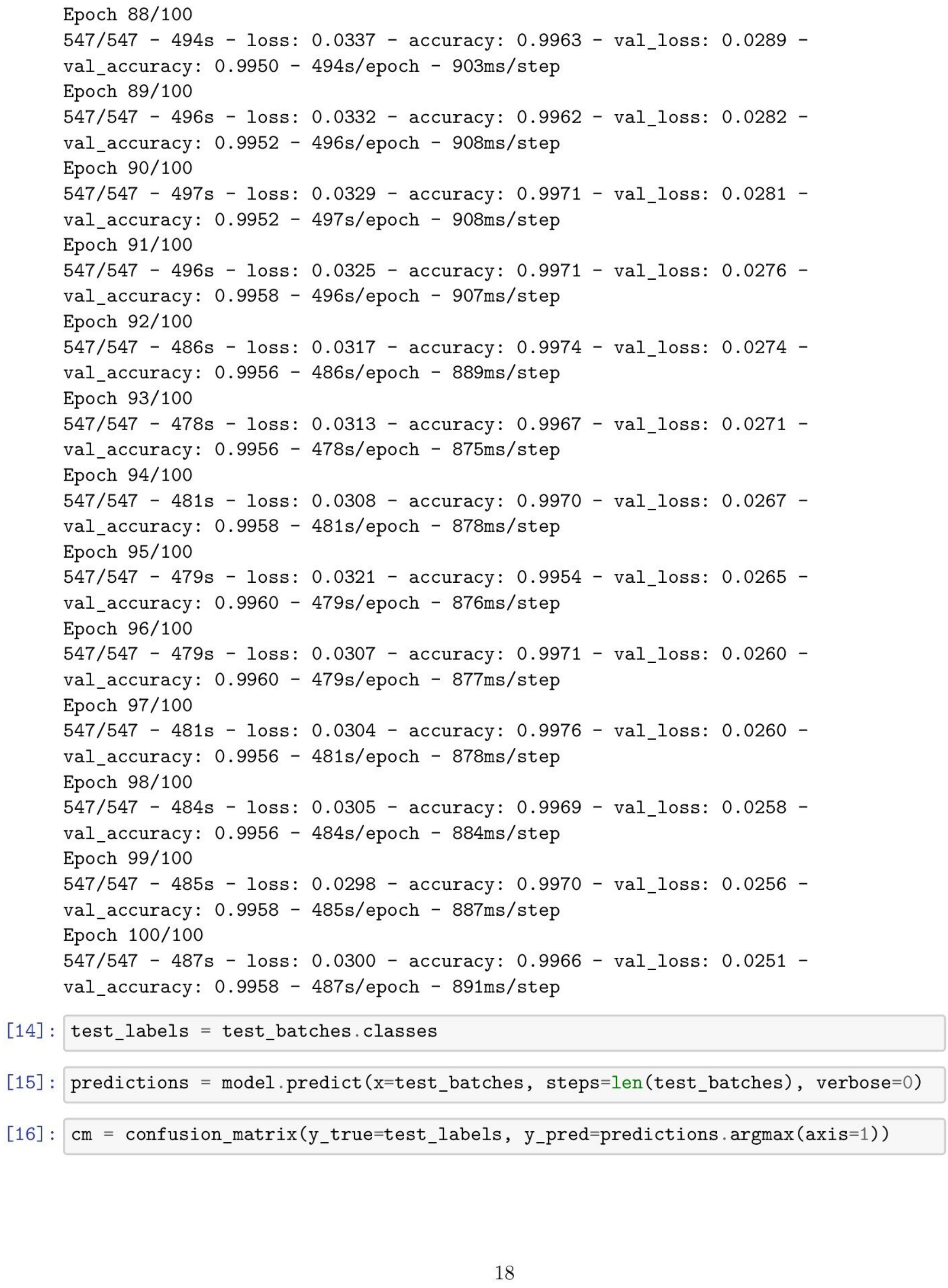


```
 conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 global_average_pooling2d_1   (None, 1, 1, 1024)       0
 (GlobalAveragePooling2D)

 reshape (Reshape)           (None, 1024)              0

 dense (Dense)               (None, 10)                10250

=================================================================
Total params: 3,239,114
Trainable params: 1,872,906
Non-trainable params: 1,366,208
_________________________________________________________________
```

```
[12]: model.compile(optimizer=SGD(learning_rate=0.0001),␣
       ↪loss='categorical_crossentropy', metrics=['accuracy'])
```

```
[13]: history=model.fit(x=train_batches,
                  steps_per_epoch=len(train_batches),
                  validation_data=valid_batches,
                  validation_steps=len(valid_batches),
                  epochs=100,
                  verbose=2
      )
```

```
Epoch 1/100
547/547 - 485s - loss: 2.0551 - accuracy: 0.3467 - val_loss: 1.3625 -
val_accuracy: 0.6120 - 485s/epoch - 886ms/step
Epoch 2/100
547/547 - 459s - loss: 1.0983 - accuracy: 0.7326 - val_loss: 0.8241 -
val_accuracy: 0.8206 - 459s/epoch - 839ms/step
Epoch 3/100
547/547 - 455s - loss: 0.7325 - accuracy: 0.8606 - val_loss: 0.5830 -
val_accuracy: 0.8926 - 455s/epoch - 832ms/step
Epoch 4/100
547/547 - 455s - loss: 0.5538 - accuracy: 0.9062 - val_loss: 0.4531 -
val_accuracy: 0.9180 - 455s/epoch - 832ms/step
Epoch 5/100
547/547 - 455s - loss: 0.4509 - accuracy: 0.9255 - val_loss: 0.3728 -
val_accuracy: 0.9332 - 455s/epoch - 831ms/step
Epoch 6/100
547/547 - 455s - loss: 0.3804 - accuracy: 0.9372 - val_loss: 0.3197 -
val_accuracy: 0.9416 - 455s/epoch - 832ms/step
Epoch 7/100
547/547 - 456s - loss: 0.3341 - accuracy: 0.9435 - val_loss: 0.2820 -
val_accuracy: 0.9484 - 456s/epoch - 833ms/step
```



```
Epoch 88/100
547/547 - 494s - loss: 0.0337 - accuracy: 0.9963 - val_loss: 0.0289 -
val_accuracy: 0.9950 - 494s/epoch - 903ms/step
Epoch 89/100
547/547 - 496s - loss: 0.0332 - accuracy: 0.9962 - val_loss: 0.0282 -
val_accuracy: 0.9952 - 496s/epoch - 908ms/step
Epoch 90/100
547/547 - 497s - loss: 0.0329 - accuracy: 0.9971 - val_loss: 0.0281 -
val_accuracy: 0.9952 - 497s/epoch - 908ms/step
Epoch 91/100
547/547 - 496s - loss: 0.0325 - accuracy: 0.9971 - val_loss: 0.0276 -
val_accuracy: 0.9958 - 496s/epoch - 907ms/step
Epoch 92/100
547/547 - 486s - loss: 0.0317 - accuracy: 0.9974 - val_loss: 0.0274 -
val_accuracy: 0.9956 - 486s/epoch - 889ms/step
Epoch 93/100
547/547 - 478s - loss: 0.0313 - accuracy: 0.9967 - val_loss: 0.0271 -
val_accuracy: 0.9956 - 478s/epoch - 875ms/step
Epoch 94/100
547/547 - 481s - loss: 0.0308 - accuracy: 0.9970 - val_loss: 0.0267 -
val_accuracy: 0.9958 - 481s/epoch - 878ms/step
Epoch 95/100
547/547 - 479s - loss: 0.0321 - accuracy: 0.9954 - val_loss: 0.0265 -
val_accuracy: 0.9960 - 479s/epoch - 876ms/step
Epoch 96/100
547/547 - 479s - loss: 0.0307 - accuracy: 0.9971 - val_loss: 0.0260 -
val_accuracy: 0.9960 - 479s/epoch - 877ms/step
Epoch 97/100
547/547 - 481s - loss: 0.0304 - accuracy: 0.9976 - val_loss: 0.0260 -
val_accuracy: 0.9956 - 481s/epoch - 878ms/step
Epoch 98/100
547/547 - 484s - loss: 0.0305 - accuracy: 0.9969 - val_loss: 0.0258 -
val_accuracy: 0.9956 - 484s/epoch - 884ms/step
Epoch 99/100
547/547 - 485s - loss: 0.0298 - accuracy: 0.9970 - val_loss: 0.0256 -
val_accuracy: 0.9958 - 485s/epoch - 887ms/step
Epoch 100/100
547/547 - 487s - loss: 0.0300 - accuracy: 0.9966 - val_loss: 0.0251 -
val_accuracy: 0.9958 - 487s/epoch - 891ms/step
```

```
[14]: test_labels = test_batches.classes
```

```
[15]: predictions = model.predict(x=test_batches, steps=len(test_batches), verbose=0)
```

```
[16]: cm = confusion_matrix(y_true=test_labels, y_pred=predictions.argmax(axis=1))
```

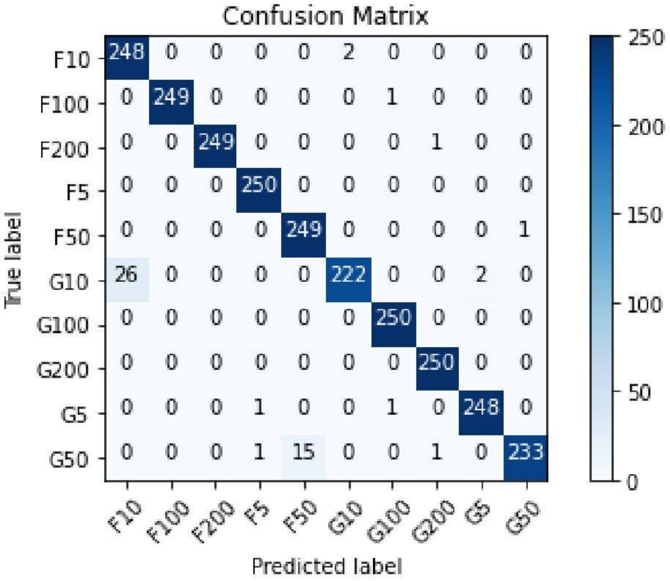
```
[17]: def plot_confusion_matrix(cm, classes,
                              normalize=False,
                              title='Confusion matrix',
                              cmap=plt.cm.Blues):
          """
          This function prints and plots the confusion matrix.
          Normalization can be applied by setting `normalize=True`.
          """
          plt.imshow(cm, interpolation='nearest', cmap=cmap)
          plt.title(title)
          plt.colorbar()
          tick_marks = np.arange(len(classes))
          plt.xticks(tick_marks, classes, rotation=45)
          plt.yticks(tick_marks, classes)

          if normalize:
              cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
              print("Normalized confusion matrix")
          else:
              print('Confusion matrix, without normalization')

          print(cm)

          thresh = cm.max() / 2.
          for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
              plt.text(j, i, cm[i, j],
                  horizontalalignment="center",
                  color="white" if cm[i, j] > thresh else "black")

          plt.tight_layout()
          plt.ylabel('True label')
          plt.xlabel('Predicted label')
```

```
[18]: test_batches.class_indices
```

```
[18]: {'F10': 0,
       'F100': 1,
       'F200': 2,
       'F5': 3,
       'F50': 4,
       'G10': 5,
       'G100': 6,
       'G200': 7,
       'G5': 8,
       'G50': 9}
```

```
[19]: cm_plot_labels =␣
        ↪['F10','F100','F200','F5','F50','G10','G100','G200','G5','G50',]
      plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='Confusion Matrix')
```

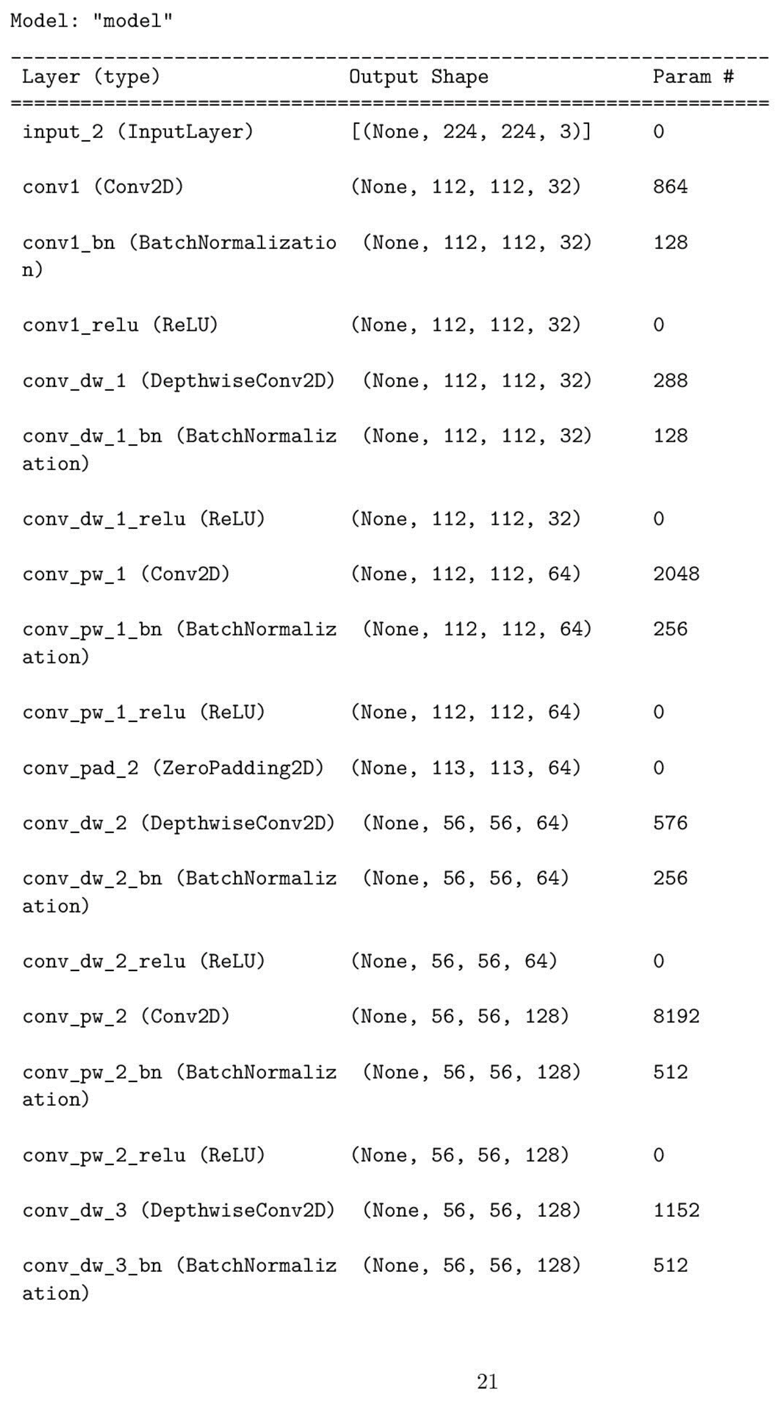
```
Confusion matrix, without normalization
[[248   0   0   0   0   2   0   0   0   0]
 [  0 249   0   0   0   0   1   0   0   0]
 [  0   0 249   0   0   0   0   1   0   0]
 [  0   0   0 250   0   0   0   0   0   0]
 [  0   0   0   0 249   0   0   0   0   1]
 [ 26   0   0   0   0 222   0   0   2   0]
 [  0   0   0   0   0   0 250   0   0   0]
 [  0   0   0   0   0   0   0 250   0   0]
 [  0   0   0   1   0   0   1   0 248   0]
 [  0   0   0   1  15   0   0   1   0 233]]
```



```
[20]: model.save('model_Result_Resnet50_SGD_00001.h5')
```

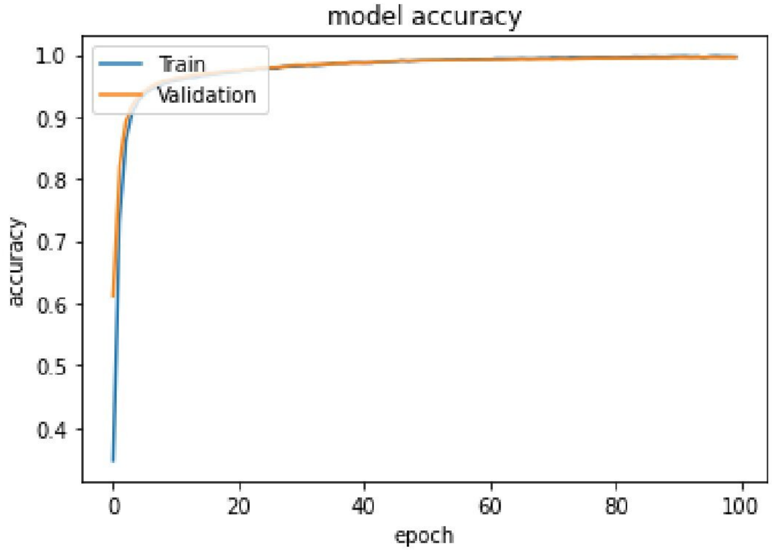
```
[21]: from tensorflow.keras.models import load_model
      new_model = load_model('model_Result_Resnet50_SGD_00001.h5')
```

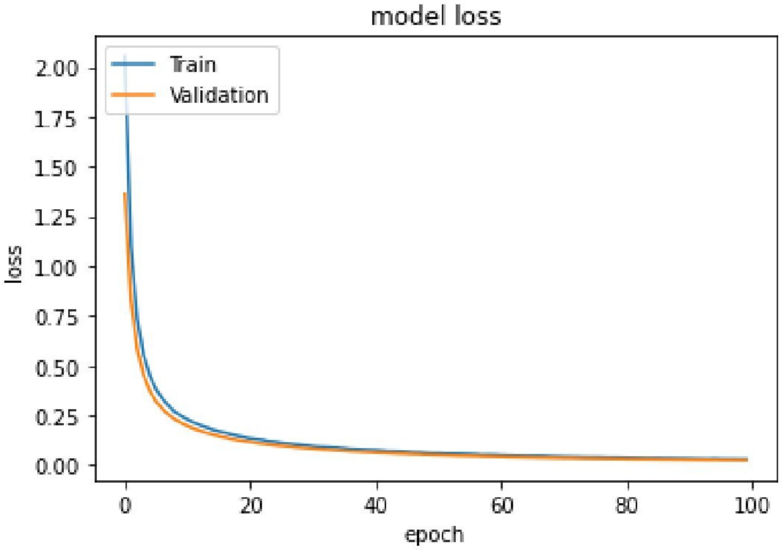
```
[22]: new_model.summary()
```



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizatio  (None, 112, 112, 32)     128
 n)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)     288

 conv_dw_1_bn (BatchNormaliz  (None, 112, 112, 32)     128
 ation)

 conv_dw_1_relu (ReLU)       (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)          (None, 112, 112, 64)      2048

 conv_pw_1_bn (BatchNormaliz  (None, 112, 112, 64)     256
 ation)

 conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0

 conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)       576

 conv_dw_2_bn (BatchNormaliz  (None, 56, 56, 64)       256
 ation)

 conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

 conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

 conv_pw_2_bn (BatchNormaliz  (None, 56, 56, 128)      512
 ation)

 conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

 conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)      1152

 conv_dw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
 ation)
 conv_dw_13 (DepthwiseConv2D  (None, 7, 7, 1024)       9216
 )

 conv_dw_13_bn (BatchNormali  (None, 7, 7, 1024)       4096
 zation)

 conv_dw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 conv_pw_13 (Conv2D)         (None, 7, 7, 1024)        1048576

 conv_pw_13_bn (BatchNormali  (None, 7, 7, 1024)       4096
 zation)

 conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 global_average_pooling2d_1   (None, 1, 1, 1024)       0
 (GlobalAveragePooling2D)

 reshape (Reshape)           (None, 1024)              0

 dense (Dense)               (None, 10)                10250

=================================================================
Total params: 3,239,114
Trainable params: 1,872,906
Non-trainable params: 1,366,208
_________________________________________________________________
```

[23]:
```python
import matplotlib.pyplot as plt

# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```


model accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
accuracy
Train
Validation
0
20
40
60
80
100
epoch


model loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
loss
Train
Validation
0
20
40
60
80
100
epoch