Developing Ethiopian Paper Currency Recognition Model Using Convolutional Neural Network (CNN)

By :Tadele Elsai Pawulos

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTING
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

OFFICE OF GRADUATE STUDIES
ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

Adama, Ethiopia
January, 2021

Developing Ethiopian Paper Currency Recognition Model Using Convolutional Neural Network (CNN)

By :Tadele Elsai Pawulos

Advisor: Tilahun Melak (PhD)

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTING
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

OFFICE OF GRADUATE STUDIES
ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

Adama, Ethiopia
January, 2021

Declaration

I, the undersigned, Student of Adama Science and Technology University, under School of Electrical Engineering and Computing, Department of Computer Science and Engineering, declare that this thesis is my original work, has not been accepted for a degree, and is not concurrently submitted in the candidature of any other degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Tadele Elsai Paulos ----------

Name Signature

Adama, Ethiopia January 2021

Place Date

This thesis work has been submitted for examination with my approval as the University Advisor.

Tilahun Melak (Ph.D.) ----------

Advisor Name Signature

Date of submission: ----------

Approval of Board of Examiners

We, the members of the Board of Examiners of the final open defense by Tadele Elsai have read and evaluated his thesis entitled "Developing Ethiopian Paper Currency Recognition Model Using Convolutional Neural Network(CNN)" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of Master in Computer Science and Engineering.

1. ---------- ---------- ----------
   Supervisor/Advisor Signature Date

2. ---------- ---------- ----------
   Chairperson Signature Date

3. ---------- ---------- ----------
   Internal Examiner Signature Date

4. ---------- ---------- ----------
   External Examiner Signature Date

5. ---------- ---------- ----------
   School Dean Signature Date

6. ---------- ---------- ----------
   School Dean Signature Date

7. ---------- ---------- ----------
   Post Graduate Dean Signature Date

Acknowledgments

First, I am most grateful to Almighty God who through his infinite mercy and love guided me throughout the program. Next, I would like to thank my advisor Dr. Tilahun Melak for providing me an opportunity to work on this project, which has been a great learning experience. I thank him for providing all the necessary resources and help to finish this study. He always encouraged me to work hard and provided all help for the completion of this study. Without his guidance, help, and support, this thesis would not have completed successfully. I appreciate his help during different phases of the study.

I would like to thank Dr. Mesfin Abebe, Department Chair of Computer Science and Engineering for his suggestions and support during my stay at ASTU. I would like to express my profound gratitude to thank Programmer Yetimeshet Tadesse for his valuable guidance in the study and for reviewing my thesis and providing valuable suggestions.

I would like to thank my family members for providing me inspiration, encouragement, and strength throughout my master's program. I would like to thank my all friends especially Kassahun Alemu, Samuel Ararsa, Muluken Yohanis, Tekeste Efirem, Zewdu Tiumay, Isak Aman, and my office staff for providing me all the necessary help to complete this research.

I am thankful to all school members of the Electrical Engineering and Computing Department for helping me to finish my CSE graduate program at Adama Science and Technology University, (ASTU).

Table of Contents

List of Figure

List of Tables

List of Equations

List of Abbreviations and Acronyms

ANN:	Artificial Neural Network
ATM:	Automatic Teller Machin
CNN:	Convolutional Neural network
Conv:	Convolutional
ConvNet:	Convolutional Neural network
DCNN:	Deep Convolutional Neural network
DL:	Deep Learning
ETB:	Ethiopian Birr
FC:	Fully connected
GA:	Genetic Algorithm
GPU:	Graphical Processing Unit
HMM:	Hidden Markov Model
AI:	Artificial Intelligence
ML:	Machin Learning
NN:	Neural Network
PCA:	Principal Component Analysis
ReLU:	Rectified Linear Unit
SGD:	Stochastic Gradient Descent

Abstract

In Ethiopia, financial institutions have adopted various transaction facilities such as banknote counting machines, multi-currency detecting machines, Automatic Teller Machines (ATMs), banknotes reader and sorters, etc. using currency recognition as their main activity, which makes automated currency recognition of significant interest. For these devices to work properly banknote recognition is a mandatory feature. In addition to these, there are 5.3% of visually impaired people in Ethiopia according to a survey report made by the Ethiopian Ministry of Health. These people have faced problems in currency recognition.

In this study, we have reviewed literature related to Ethiopian and other country's banknotes recognition. In the case of Ethiopian banknote recognition and classification, there is little research work on old Ethiopian banknotes. There have not been found any methods implemented or proposed for the recognition of newly released Ethiopian banknotes. Therefore, this thesis presents a deep learning-based method for newly released Ethiopian paper currency denominations from their color images. A classification framework has been implemented using the concept of transfer learning where a large convolutional neural network pre-trained on millions of natural images is employed for the classification of images from new classes. An image dataset of four banknote denominations is prepared by preprocessing and augmentation of real-banknote images acquired in different viewpoints and lighting conditions via a scanner and smartphone camera. A new top layer upon the convolutional base of a pre-trained Mobile Net model is trained for a few epochs upon a portion of the dataset to achieve an agreeable accuracy. All the codes are implemented using Python programming language. In the case of the 80/10/10 dataset proportion an average Training accuracy and average validation accuracy of 99.7%, 99.5% are obtained respectively, and in the case of 70/15/15 dataset proportion an average Training accuracy and average validation accuracy of 97.8%, 99.0% are obtained respectively. Therefore, as the first research work, the model has shown the best performance in both dataset categories but the 80/10/10 dataset ratio provides a promising result with an average training accuracy of 99.7%.

Keywords: *Currency*, *Recognition, Deep Learning, Neural Networks, Transfer Learning*

CHAPTER ONE: INTRODUCTION

1.1. Background of the Study

Nowadays, Machine Learning (ML) has a profound influence on a wide range of applications from text understanding, image and speech recognition, to health care, and many more. It is so ubiquitous that there is hardly anyone with access to technology today untouched by machine learning [1]. They are widely applied in a variety of engineering areas such as pattern recognition, natural language processing, and computational learning [2]. Recently a specialized domain from machine learning called Deep Learning (DL) has emerged out to yield efficient techniques for tasks of object classification or recognition. It has attracted the attention of many high-tech enterprises such as Google, Facebook, and Microsoft. Most of the frontrunners of image processing competitions are employing deep CNN-based models today. It has been widely adopted in various directions of computer vision, such as image classification, object detection, image retrieval and semantic segmentation, and human pose estimation, which are key tasks for image understanding [3]. Also, it has been applied to paper currency recognition, classification, and counterfeit detection, etc.

Paper currency recognition and classification using neural networks (NNs) is an important area of pattern recognition that attracts the attention of different researchers today. It is a highly demanded area of feature extraction. A system for the recognition of paper currency is one kind of intelligent system which is a very important need of the current automation systems. In this age of scientific advancement, automated machinery for handling banknotes plays a significant role in our daily life as financial institutions perform thousands of monetary transactions every day. Furthermore, the improvement in modern-day banking operations and automated systems for the identification of currencies have become pertinent in many applications such as electronic banking, currency monitoring system, money exchange machine, Automated Teller machines (ATM), and vending and parking meters [4]. Despite considerable technological advancements in the field of plastic money and e-commerce, modern-day monetary transactions involve the massive flow of paper-based currency. Specifically in Ethiopia, still hand to hand cash transaction is widely used for day to day routines. Ethiopia has both paper and coin currencies. The name of the Ethiopian paper

currency is called ("Birr = ETB").There are 6 types that are 1ETB, 5ETB, 10ETB, 50ETB, 100ETB, and 200ETB [5]. Each of them has a specific color, size, security, and identification features. Among all types, the newly released version was used in this thesis. *[See Appendix A]*

Paper currency recognition and sorting system are important for not only the efficient running of currency handling machines such as ATMs, vending machines, etc., they can also be used in devices for visually challenged persons. Eye vision problems are considered as one of the major public health issues in the Africa continent. The WHO report indicates that 90% of the blind peoples were living in developing countries [6]. According to the national survey of the Ethiopian Ministry of Health along with several non-governmental organizations in 2006 G.C, there were about 5.3% of the total population of the country lives with blindness and low vision problems (1.6% blind and 3.7% low vision) [7]. So in addition to the need for automation, banknote recognition may assist visually impaired people in terms of better decisions of denominations inscribed on banknotes and also involve in socio-economic activities of their country.

Currently, there are various currency recognition systems and devices available internationally. Some of which support multiple currencies while others are currency dependent. Currency Recognition systems are built behind the extraction of sufficient, stable, and distinctive monetary features significant for accuracy and robustness from the respective currency using various recognition algorithms [8]. In the context of denomination recognition and classification from different countries banknotes images, several methods have been proposed earlier by several researchers. For example, Convolutional Neural Network (CNN), Hidden Markov Model (HMM), Genetic Algorithm (GA), Artificial Neural Network (ANN), local feature descriptor comparison, Principal Component Analysis (PCA), and Naïve Bayes classifier are among the various techniques used. CNN is applied in Indian banknotes recognition [9], and New Zealand currency recognition [10]. ANN is applied in Thai banknote recognition [11], and Indian currency recognition [12]. Local feature descriptor comparison is applied in Ethiopian paper currency recognition [13]. Also, other approaches are discussed in the next chapter two.

Also, there are very little researches that have been done on Ethiopian paper currency recognition and classification, and counterfeit detection using deep learning. These researches were done on old Ethiopian banknote images using different recognition techniques [13,14,15]. However, there is no research done on newly released Ethiopian banknote image data or there is no machine learning model that enables to recognize and classify newly released Ethiopian paper currency. So the major aim of this study is to prepare a labeled custom dataset for Ethiopian banknotes and develop a machine learning model that enables the classification of newly issued Ethiopian banknotes. The model utilizes the concept of transfer learning where a deep convolutional neural network already trained upon a huge dataset of natural images is re-utilized for the problem of classification of the denomination from banknote images. The real images of banknotes taken under variable lighting conditions and different viewing perspectives are fed to a custom-made neural network topped over the pre-trained convolutional base to learn the new classes associated with the problem. Hence obtained classifier trained upon a modestly sized dataset, achieve considerably good accuracy. The method requires the least preprocessing of images while feeding to the classifier and works well for recognition and classification even in presence of background clutter.

1.2. Motivation

As we all know, the numbers of financial institutions like banks and credit unions have increased in Ethiopia today. Due to this reason, the numbers of automatic transaction facilities such as banknote counting machines, multi-currency detecting machines, Automatic Teller Machines (ATMs), banknotes reader and sorters, etc. have increased dramatically. All these devices have been imported from abroad. For example, the Commercial Bank of Ethiopia (CBE) has invested a huge amount of money to import these devices. Currently, I am working at CBE as a senior office and banking automation engineer. What we have observed is that these devices have a problem recognizing and classify Ethiopian paper currency denominations properly. Most of them are currency-dependent. They do not consider Ethiopian banknote conditions when designed and manufactured. As far as I know, there is no Ethiopian banknote value recognition and fitness classification machine. Banknotes classification has been done manually in CBE. A huge amount of manpower was employed to sort millions of banknotes which is time-consuming. Also, no portable banknote recognition devices that can help visually impaired people in Ethiopia.

In addition to these, there are several types of research works done in the area of paper currency recognition, counterfeit detection, and fitness classification in different parts of the world including Ethiopia [13,14,15]. However, no research was done on newly released Ethiopian Birr denominations. Therefore, all these are why we need to develop a deep learning model that can recognize and classify newly released Ethiopian paper currency denominations (10 ETH, 50 ETB, 100 ETB, and 200 ETB) effectively and efficiently.

1.3. Statement of the Problem

Paper currency recognition with good accuracy and high processing speed has great importance for the bank system, automated selling- systems, and assisting visually impaired people. Today, how to extract high-quality monetary features from currency images is a key problem area that needs solutions in paper currency recognition. In this regard, in a different part of the world, different researchers have been doing research and suggest a software and hardware solution in recognizing paper currency denominations. To mention some of the studies which contributed to this concern:

Automated Bank Note Identification System for Visually Impaired Subjects in Malaysia [8], Indian banknote recognition using convolutional neural network on for Indian Rupee [9], Thai banknote recognition using neural network and continues learning by DSP unit[11], Mexican banknotes recognition and classification via their color and texture features [16,17], In Saudi Arabian on an intelligent paper currency recognition system using Radial Basis Function network classifier [18]. And also in Ethiopia, automatic recognition and counterfeit detection system were done on old Ethiopian banknote [13,14,15]. The solution for the problem of Ethiopian old banknote recognition and classification has been done so far by using different banknote recognition methods. However, there is no research done on newly released Ethiopian paper currency. The classification of these newly released Ethiopian banknotes is carried out manually which makes the bank employee busy. It is consuming a lot of time. There are no newly released Ethiopian banknotes sorter machines or model, and custom datasets. Commercial Bank of Ethiopia (CBE) invested huge foreign currency to purchases machines that can recognize and

classify Ethiopian Birr. But these machines have a problem with ETB recognition and classification.

Additionally, According to the national survey of the Ethiopian Ministry of Health along with several non-governmental organizations in 2006 G.C, there were about 5.3% of the total population of the country lives with blindness and low vision problems (1.6% blind and 3.7% low vision)[7]. These visually impaired peoples are confronted with many challenges during their daily lives especially when conducting financial transactions. Although there are several traditional techniques exist that assist in identifying the different denominations of the banknote, they also have their problems and limitations to their effectiveness. However, overtime banknotes get old and the line could get blurry making it difficult for visually impaired people to accurately identify the different banknote denominations. So, the main concern of this study is to find solutions for the above-mentioned problems by developing an efficient Deep Learning model that can recognize and classify Ethiopian newly released paper currency.

1.4. Research Questions

The questions that this research is going to answer are as follows:

RQ1: Which image processing technique is suitable for recognizing newly released Ethiopian Paper Currency denominations?

RQ2: How to create custom datasets that appropriately train and test the model created?

RQ3: How to train the model with created custom datasets and different learnable parameters values?

RQ4: What is the efficiency of the model when tested?

1.5. Objectives

1.5.1. General Objective

The general objective of this study is to develop a CNN-based model that enables us to recognize newly-released Ethiopian Paper Currency.

1.5.2. Specific Objectives

To achieve the general objective of the study, the following specific objectives are formulated:

- In-depth literature study on background knowledge such as convolutional neural network algorithms, Ethiopian currency banknotes, different countries' approaches to recognize banknotes, and other related concepts.
- Custom dataset preparation for training, validation, and testing the model.
- Develop the recognition model using a pre-trained CNN model.
- Evaluate the performance of the developed model.
- Draw conclusions and forward recommendations.

1.6. Significance of the Study

The study provides various benefits for visually impaired people, different financial institutions such as banks and other researchers. It can be applied as a software solution in a different area:

- Financial institutions can be a beneficiary of the study.
- It opens the door for different researchers who are interested to research Ethiopian banknotes recognition.
- This work will be a software component for implementing portable embedded paper currency recognition devices and an application assisting visually impaired people. Since the proposed work will recognize banknote, it can be a great input and solution to the

bank industry, visually impaired people, and other society for their daily financial transactions.

1.7. Scope of the Study

The study focuses on developing a model that can recognize all newly released Ethiopian banknotes denominations (such as 10 Birr, 50 Birr, 100 Birr, and 200 Birr) from all its distinct images. The counterfeit detection and fitness classification are not included in this study.

1.8. Limitations of the Study

The limitation of the study is the small size of the datasets used and computational power. So with this small amount of data, it is impossible to achieve a CNN model with high efficiency even if this study has achieved better efficiency. Therefore, computational power and dataset size are the limitations of this research.

1.9. Organization of the Study

This thesis is structured into seven chapters. Chapter One presents an introduction that shows the background and motivation of the study. Chapter Two presents relevant and related literature reviews on important concepts and elements that were used in the thesis. Machine Learning, Deep Learning, computer vision, paper currency recognition, and convolution neural network are discussed. It also includes related works done by different researchers. Chapter Three presents the methodology we have used to accomplish the research work. Datasets preprocessing and augmentation were also discussed. Chapter Four presents the proposed model implementation and dataset preparation. The architecture of the model is also included in detail. Chapter Five presents the results we got on the model and discusses in detail why and how we got those results. Lastly, Chapter Six presents conclusions and recommendations of the study.

CHAPTER TWO: LITERATURE REVIEW

2.1. Currency and Paper Currency Recognition Approaches

2.1.1. Currency

Currency is started serving as a medium for exchanging goods and services thousands of years ago to replace an ancient barter system where any objects could be swapped if the traders agreed. It is an indispensable part of our daily life. It may take a physical form as coins and banknotes or may exist as a written or electronic account. Despite the decreasing use of paper currency and rapidly increasing electronic transactions, paper currencies remain essential because they are easily carried, widely accepted, and reliable. Currently, there are over 200 different currencies with different characteristics are used around the globe [12]. So that currency recognition technology aims to search and extract paper currency features for effective recognition and classification.

2.1.2. Paper Currency Recognition Approaches

Paper currency recognition aims to determine the denomination and orientation of banknotes. It's used in several areas, including banks, companies, railways, shopping malls, department stores, and government departments. For monetary transactions, devices such as banknote counting, sorting, and automated teller machines [ATMs] must accurately identify and distinguish banknote denominations with high-speed. The following measures are followed by the majority of paper currency recognition methods:

Banknote images are first obtained using a scanner or camera. Preprocessing and feature extraction are then used to extract useful information from the cropped banknote images. Finally, classification is carried out using the extracted features to identify the banknote's course and to distinguish banknotes of different denominations [19]. Conventional or traditional paper currency recognition methods use digital image processing techniques such as wavelet transforms or SVM to acquire characteristics for classification. According to recent studies, machine learning

techniques, such as Convolutional Neural Networks (CNNs), can extract useful features without the use of handcrafted methods in contrast to digital image processing systems. CNN-based techniques like Google Net, Dense Net, and Mobile Nets performed extremely well in image classification tasks [20] .

2.2. Overview of Machine Learning

The roots of Machine Learning date back to the early 1950s, not long after the invention of the first general-purpose electronic computer, when the first Turing test was developed by Alan Turin [21], to decide if a computer can think or to put it more accurately whether a machine can learn. Doctor Turing thought that it would be possible for given time machines to "think" and "learn" and ultimately pass the Turing Test. Working on this theory, many scientists began to work on this "Artificial Intelligence" principle and in 1952, Arthur Samuel wrote the first checkers software that learned and adapted its strategy by playing against human opponents and was finally able to defeat human players in the early 1970s. The term Artificial Intelligence (AI) later evolved into what we now know as Machine Learning. Figure 2-1 provides a view of the Venn diagram of how all these words are inter-related.

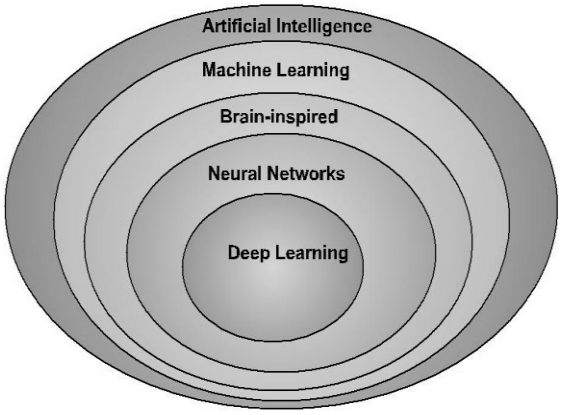


Figure 2. 1: Artificial Intelligence, Machine and Deep Learning (Taken from [21])

In [22], machine learning is described as "A computer program is said to learn from experience E with -respect- to some task T and some performance measure P if its performance on T improves with experience E as measured by P."

In the years after the initial concept, a lot of work has been done, with numerous algorithms being discovered [23] and fine-tuned. The biggest advance in neural networks came after researchers rediscovered the back-propagation algorithm and its benefits for the entire field of machine learning. Deep learning is a term coined in 2006 to describe neural networks that are modeled after the human brain. As compared to the more conventional knowledge-driven approaches, neural networks use a more data-driven approach. Computers can now "see" images and videos, as well as discriminate between objects and documents, thanks to modern Deep Learning algorithms.

2.3. Supervised Type of Machine Learning

Supervised Learning is a Machine Learning algorithm in which training data comprises an example of input vectors along with their corresponding target vectors. These are for solving regression and classification problems. Image classification is a supervised learning problem in which the algorithm is given labeled training data to learn from. The algorithm can then use the previously trained data set to predict future occurrences of the mark in question. Figure 2-2 illustrates a supervised learning pipeline. It's important to note that while the training part of the algorithms differs (determining the weights), the inference/prediction part (applying weights to determine output) is consistent across all of them.

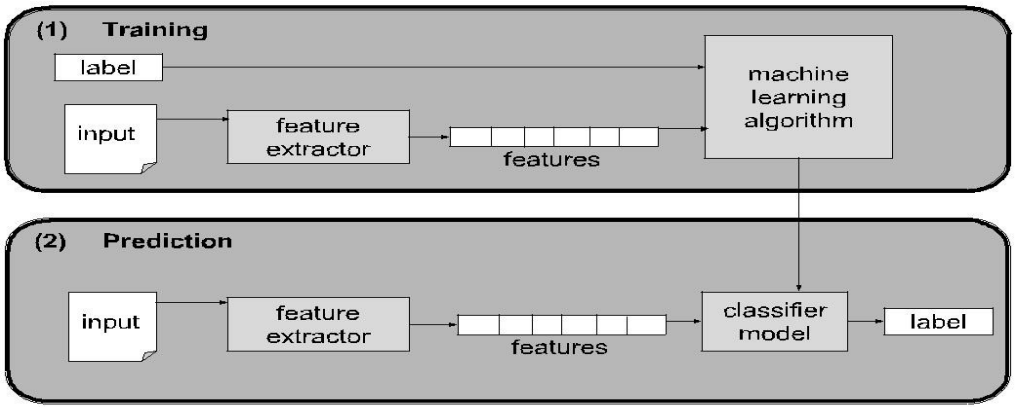


Figure 2. 2: Generic Supervised Machine Learning Pipeline

2.4. Machine Learning for image classification

Since the beginning of machine learning, image recognition has been a key field of research because it is the primary way for computers to interact with our physical environment. The first step for machines to achieve actionable intelligence is for them to recognize and remember objects in pictures. Single-layer systems were used to extract features and classify images at the start of Machine Learning-based approaches to solving the classification problem, but the study came to a halt in the late 1960s due to problems with accuracy and implementation of such systems. Convolutional neural networks (CNNs), developed at Bell Labs in 1988, were the most important development in supervised learning. They rapidly gained popularity, and CNN's were used in several systems, including the check reading method that was implemented in the United States and Europe in 1996.

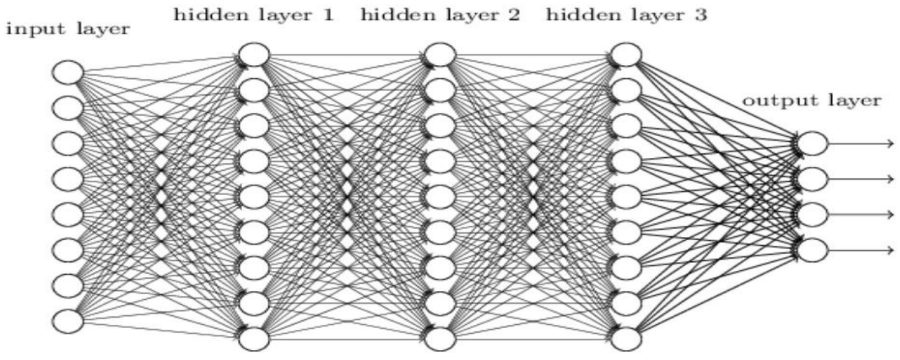


Figure 2. 3: Multi-layer neural networks

CNNs are a form of multi-layer neural network (shown in Figure 2-3) that can span multiple layers to improve accuracy and thus require a lot of computing power. CNN that span multiple layers (more than three) are referred to as Deep Learning (DL) networks. These DL networks needed a lot of computing resources, which were scarce in the early 2000s. But, in 2012, people discovered that you could get inexpensive supercomputers in the form of GPUs that could do Teraflops. After the University of Toronto (UOT) study [24] using these GPUs, acceptance skyrocketed as error rates fell drastically. Today, whether you use a CNN, you won't be able to get your research written.

2.5. Computer Vision

These days, machine vision has a wide variety of applications, from factory optical inspection to Facebook's auto-tag neural network, which processes billions of pictures per day [25]. The problem of image recognition and object detection can be solved in many ways using clustering, Principal Component Analysis (PCA), Support Vector Machines (SVM), and Neural Networks. Of all the visual tasks we could ask a computer to perform, analyzing a scene and identifying all the objects remains the most challenging. In this study category of the neural network called CNN is used. The neural network is a brain-inspired network that uses neurons to classify and identify images. Among neural networks, the most promising is the Deep Learning technology which is based on artificial neural networks composed of many layers. Based on the power of these multi-layers CNN's, this category of Machine Learning revolutionizes computer vision applications, by offering high recognition rates and robustness needed by many critical applications.

2.6. Why Deep Learning?

Deep Learning is a subset of machine learning with more power and flexibility than traditional machine learning approaches. The biggest advantage of Deep Learning algorithms is that they try to learn high-level features from data in an incremental manner. This prevents the need for manual feature extractions which is done by human intervention in traditional machine learning. Deep learning emulates the human brain [26], and it works with unstructured data, especially for applications like computer vision (image processing) and speech recognition. It performs more and more when we have a large data size and good computational performance. It outperforms other approaches when it comes to complex problems like image classifications, natural language processing, and speech recognition. We need to have two crucial things in Deep Learning. The first one is large data size to train our model and the second one is a good computational device because deep learning learns up to millions of parameters in a model and our device should be capable of overcoming those bunches of parameters easily.

2.7. Convolutional Neural Networks

CNN is a class of deep learning model most commonly used for processing data with a grid pattern, such as images, that is inspired by the organization of the animal visual cortex [27,28] and designed to learn spatial hierarchies of features, from low- to high-level patterns, automatically and adaptively. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected (Dense) layers. The first two, convolution and pooling layers, perform feature extraction, while the third, a fully connected layer, maps the extracted features into the final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers and a small grid of parameters called the kernel, and an optimizable feature extractor is applied at each image position, which makes CNN's highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

2.7.1. Why is CNN different from other methods in Banknotes Recognitions?

Most recent paper currencies studies use hand-crafted feature extraction techniques, such as texture analysis, followed by conventional machine learning classifiers, such as random forests and support vector machines [29, 30]. There are several differences to note between such methods and CNN. First, CNN does not require hand-crafted feature extraction. Second, CNN architectures do not necessarily require the segmentation of banknote features by human experts. Third, CNN is far more data-hungry because of its millions of learnable parameters to estimate, and, thus, is more computationally expensive, resulting in requiring graphical processing units (GPUs) for model training.

2.7.2. Building blocks of CNN architecture

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A typical architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. The step where input data are transformed into output through these layers is called forward propagation (Fig.2.4). Although convolution and pooling operations described in this section are for 2D-CNN, similar operations can also be performed for three-dimensional (3D)-CNN.

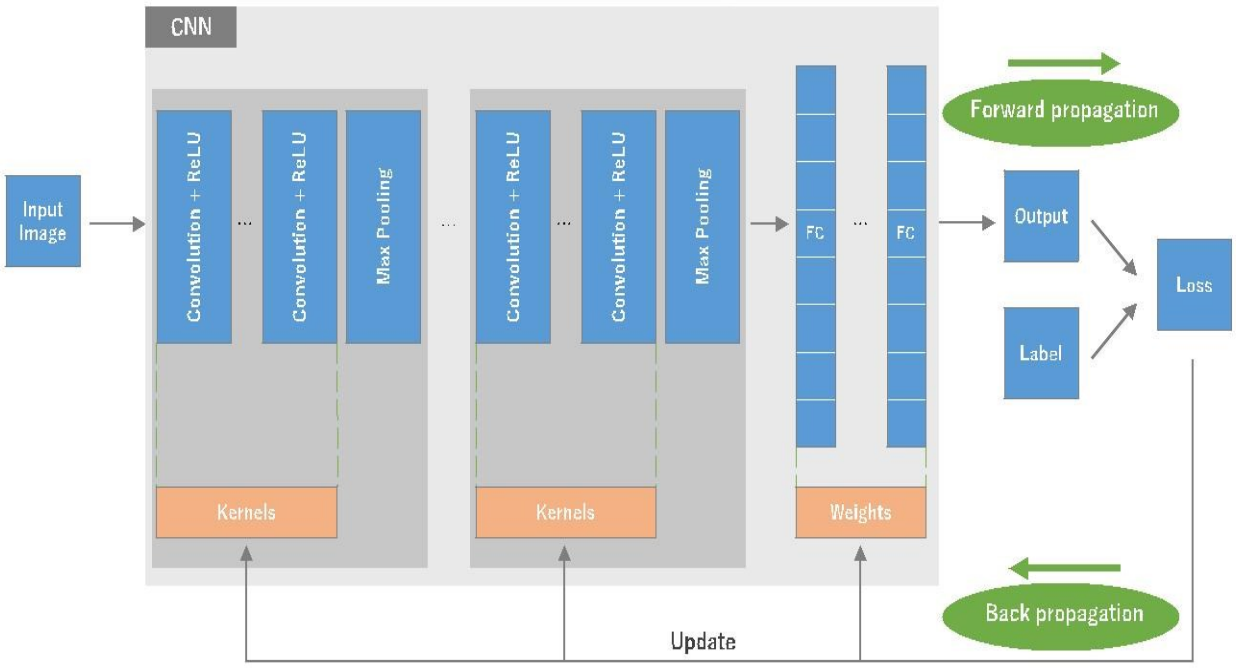


Figure 2. 4: CNN Architecture and Training Process (Taken from [35]

Figure 2.4 above shows an overview of the convolutional neural network (CNN) architecture and the training process. A CNN is composed of stacking of several building blocks: convolution layers, pooling layers (e.g., max-pooling), and fully connected (FC) layers. A model's performance under particular kernels and weights is calculated with a loss function through forwarding propagation on a training dataset, and learnable parameters, i.e., kernels and weights are updated according to the loss value through backpropagation with gradient descent

optimization algorithm and ReLU (Rectified linear unit).

2.7.3. Convolution Neural Network (CNN) Layers

2.7.3.1. Convolution Layer

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, i.e., convolution operation and activation function. The convolution operation allows the center of each kernel to overlap the outermost element of the input tensor and reduces the height and width of the output feature map compared to the input tensor. Padding, typically zero paddings, is a technique to address this issue, where rows and columns of zeros are added on each side of the input tensor, to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation. Modern CNN architectures usually employ zero paddings to retain in-plane dimensions to apply more layers. Without zero paddings, each successive feature map would get smaller after the convolution operation.

The distance between two successive kernel positions is called a stride, which also defines the convolution operation. The common choice of a stride is 1; however, a stride larger than 1 is sometimes used to achieve downsampling of the feature maps. An alternative technique to perform downsampling is a pooling operation, as described below. The key feature of a convolution operation is weight sharing: kernels are shared across all the image positions. Weight sharing creates the following characteristics of convolution operations: (1) letting the local feature patterns extracted by kernels translation b invariant as kernels travel across all the image positions and detect learned local patterns,(2) learning spatial hierarchies of feature patterns by downsampling in conjunction with a pooling operation, resulting in capturing an increasingly larger field of view, and (3) increasing model efficiency by reducing the number of parameters to learn in comparison with fully connected neural networks.

As described later, the process of training a CNN model about the convolution layer is to identify the kernels that work best for a given task based on a given training dataset. Kernels are the only

parameters automatically learned during the training process in the convolution layer; on the other hand, the size of the kernels, number of kernels, pad-ding, and stride are hyperparameters that need to be set before the training process starts (Table 2.1).

Table 2. 1: A list of parameters and hyperparameters in a convolutional neural network (CNN)

Layers	Parameters	Hyperparameters
Convolution layer	Kernels	Kernel size, number of kernels, stride, padding, activation function
Pooling layer	None	Pooling method, filter size, stride, padding
Fully connected layer	Weights	Number of weights, activation function
Others		Model architecture, Optimizer, learning rate, loss function, mini-batch size, epochs, regularization, weight initialization, dataset splitting

2.8. Nonlinear activation function

The outputs of a linear operation such as convolution are then passed through a nonlinear activation function. Although smooth nonlinear functions, such as sigmoid or hyperbolic tangent (tanh) function, were used previously because they are mathematical representations of a biological neuron behavior, the most common nonlinear activation function used presently is the rectified linear unit (ReLU), which simply computes the function:

F(x) = max (0, x)……. (2.1)

Equation 2. 1:ReLU Equation

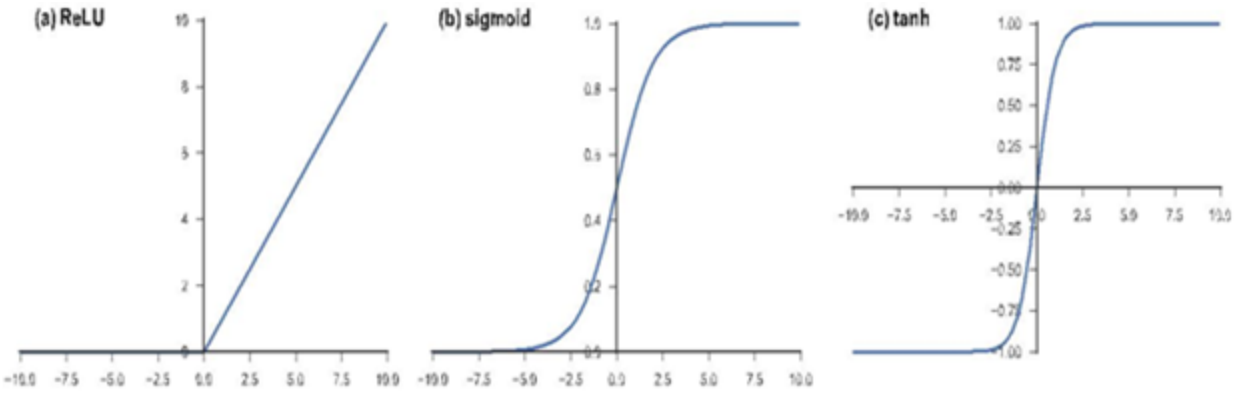


Figure 2. 5: Activation functions commonly applied to neural networks: (a) rectified linear unit (ReLU), (b) sigmoid, and (c) hyperbolic tangent (tanh).

2.9. Pooling Layer

A pooling layer provides a typical downsampling operation which reduces the in-plane dimensionality of the feature maps to introduce translation invariance to small shifts learnable parameters. It is of note that there is no learnable parameter in any of the pooling layers, whereas filter size, stride, and padding are hyperparameters in pooling operations, similar to convolution operations.

2.9.1. Max - Pooling

The most popular form of pooling operation is max pooling, which extracts patches from the input feature maps, outputs the maximum value in each patch, and discards all the other values. A max-pooling with a filter of size 2 × 2 with a stride of 2 is commonly used in practice. This downsamples the in-plane dimension of feature maps by a factor of 2. Unlike height and width, the depth dimension of feature maps re-mains unchanged.

2.9.2. Global average pooling

Another pooling operation worth noting is a global aver-age pooling [30]. A global average pooling performs an extreme type of downsampling, where a feature map with a size of height × width is downsampled into a 1 × 1 array by simply taking the average of all the elements in each feature map, whereas the depth of feature maps is retained. This operation is typically applied

only once before the fully connected layers. The advantages of applying global average pooling are as follows: (1) reduces the number of learnable parameters and (2) enables the CNN to accept inputs of variable size.

2.10. Fully connected layer

The output feature maps of the final convolution or pooling layer are typically flattened, i.e., transformed into a one-dimensional (1D) array of numbers (or vector), and connected to one or more fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. Once the features extracted by the convolution layers and downsampled by the pooling layers are created, they are mapped by a sub-set of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks. The final fully connected layer typically has the same number of output nodes as the number of classes. Each fully connected layer is followed by a non-linear function, such as ReLU, as described above.

2.11. Activation function

The activation function applied to the last fully connected layer is usually different from the others. An appropriate activation function needs to be selected according to each task. An activation function applied to the multiclass classification task is a soft-max function that normalizes out-put real values from the last fully connected layer to target class probabilities, where each value ranges between 0 and 1 and all values sum to 1. For this study, Soft-max is used. Typical choices of the last layer activation function for various types of tasks are summarized in Table 2. 2.

Table 2. 2: A list of commonly applied last layer activation functions for various tasks

Task	Last layer activation function
Binary classification	Sigmoid
Multiclass single-class classification	Soft-max
Multiclass multi-class classification	Sigmoid
Regression to continuous values	Identity

2.12. Training a network

Training a network is a process of finding kernels in convolution layers and weights in fully connected layers which minimize differences between output predictions and given ground-truth labels on a training dataset. The backpropagation algorithm is the method commonly used for training neural networks where loss function and gradient descent optimization algorithm play essential roles. A model performance under particular kernels and weights is calculated by a loss function through forwarding propagation on a training dataset, and learnable parameters, namely kernels and weights are updated according to the loss value through an optimization algorithm called back-propagation and gradient descent, among others (Fig. 2.4).

2.13. Loss function

A loss function, also referred to as a cost function, measures the compatibility between output predictions of the network through forwarding propagation and given ground truth labels. A commonly used loss function for multiclass classification is cross-entropy, whereas mean squared error is typically applied to regression to continuous values. A type of loss function is one of the hyperparameters and needs to be determined according to the given tasks.

2.14. Gradient descent

Gradient descent is commonly used as an optimization algorithm that iteratively updates the learnable parameters, i.e., kernels and weights, of the network to minimize the loss. The gradient of the loss function provides us the direction in which the function has the steepest rate of

increase, and each learnable parameter is updated in the negative direction of the gradient with an arbitrary step size determined based on a hyper-parameter called learning rate (Fig.2.6). The gradient is, mathematically, a partial derivative of the loss concerning each learnable parameter, and a single update of a parameter is formulated as follows:

$$w := w - \alpha * \frac{\partial L}{\partial w} \quad \text{.....(2.2)}$$

Equation 2. 2:Gradient descent Partial derivative equation

, Where **w** stands for each learnable parameter, **α** stands for a learning rate, and **L** stands for a loss function. It is of note that, in practice, a learning rate is one of the most important hyper-parameters to be set before the training starts. In practice, for reasons such as memory limitations, the gradients of the loss function about the parameters are computed by using a subset of the training dataset called mini-batch and applied to the parameter updates. This method is called mini-batch gradient descent, also frequently referred to as stochastic gradient descent (SGD), and a mini-batch size is also a hyper-parameter. Also, many improvements on the gradient descent algorithm have been proposed and widely used, such as SGD with momentum, RMSprop, and Adam [31, 32, and 33]. Adam optimizer is used for our study.

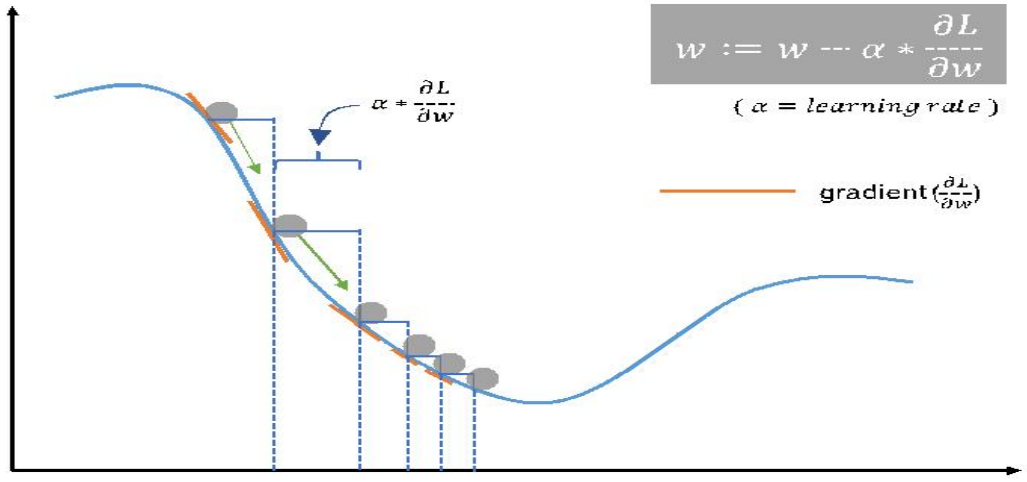


Figure 2. 6: Gradient descent

2.15. Data and ground truth labels

Data and ground truth labels are the most important components in research applying deep learning or other machine learning methods. As a famous proverb originating in computer science notes: "Garbage in garbage out.", careful collection of data and ground truth labels with which to train and test a model is mandatory for a successful deep learning project, but obtaining high-quality labeled data can be costly and time-consuming.

Available data are typically split into three sets: training, validation, and a test set (Fig. 8), though there are some variants, such as cross-validation. A training set is used to train a network, where loss values are calculated via forwarding propagation and learnable parameters are updated via back-propagation. A validation set is used to evaluate the model during the training process, fine-tune hyper-parameters, and perform model selection. A test set is ideally used only once at the very end of the project to evaluate the performance of the final model that was fine-tuned and selected on the training process with training and validation sets.

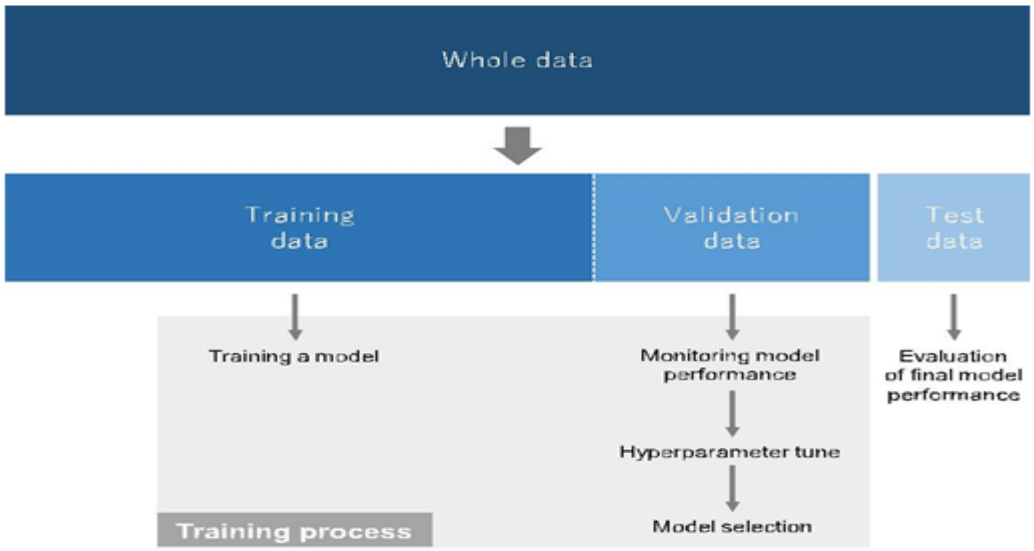


Figure 2. 7: Available data split into Training data, Validation data, and Test data

Separate validation and test sets are needed because training a model always involves fine-tuning its hyper-parameters and performing the model selection. As this process is performed based on the performance of the validation set, some information about this validation set leaks into the model itself, i.e., overfitting to the validation set, even though the model is never directly trained on it for the learnable parameters. For that reason, it is guaranteed that the model with fine-tuned hyperparameters on the validation set will perform well on this same validation set. Therefore, a completely unseen dataset, i.e., a separate test set, is necessary for the appropriate evaluation of the model performance, as what we care about is the model performance on never-before-seen data, i.e., generalizability.

It is worthy to mention that the term "validation" is used differently in the medical field and the machine learning field [31]. As described above, in machine learning, the term "validation" usually refers to a step to fine-tune and select models during the training process. On the other hand, "validation" usually stands for the process of verifying the performance of a prediction model, which is analogous to the term "test" in machine learning. To avoid this confusion, the word "development set" is sometimes used as a substitute for "validation set".

2.16. Overfitting

Overfitting refers to a situation where a model learns statistical regularities specific to the training set, i.e., ends up memorizing the irrelevant noise instead of learning the signal, and, therefore, performs less well on a subsequent new dataset. This is one of the main challenges in machine learning, as an over-fitted model is not generalizable to never-seen-before data. In that sense, a test set plays a pivotal role in the proper performance evaluation of machine learning models, as discussed in the previous section. A routine check for recognizing overfitting to the training data is to monitor the loss and accuracy of the training and validation sets (Fig.2.9).

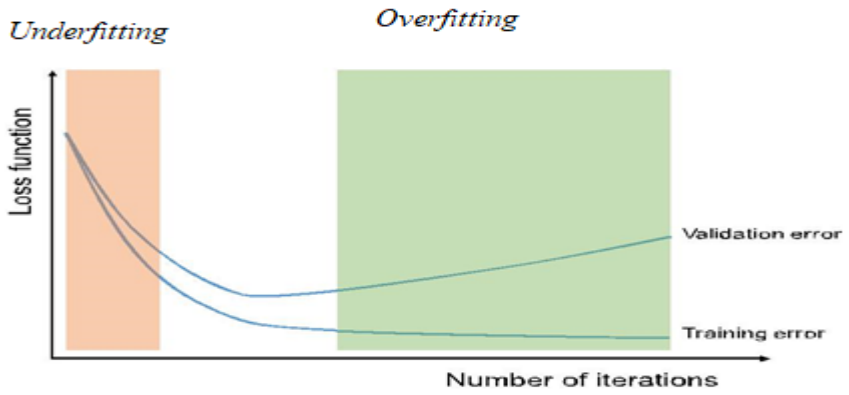


Figure 2. 8: shows overfitting and underfitting

If the model performs well on the training set compared to the validation set, then the model has likely been overfitting to the training data. There have been several methods proposed to minimize overfitting (Table 2. 3). The best solution for reducing overfitting is to obtain more training data. A model trained on a larger dataset typically generalizes better. The other solutions include regularization with dropout or weight decay, batch normalization, and data augmentation, as well as reducing architectural complexity. Dropout is a recently introduced regularization technique where randomly selected activations are set to 0 during the training so that the model becomes less sensitive to specific weights in the network [32]. Weights decay, also referred to as L2 regularization reduces overfitting by penalizing the model's weights so that the weights take only small values. Batch normalization is a type of supplemental layer that adaptively normalizes the input values of the following layer, mitigating the risk of overfitting, as well as improving gradient flow through the network, allowing higher learning rates, and reducing the dependence on initialization [33]. Data augmentation is also effective for the reduction of overfitting, which is a process of modifying the training data through random transformations, such as flipping, translation, cropping, rotating, and random erasing so that the model will not see the same inputs during the training iterations [34]. Despite these efforts, there is still a concern of overfitting to the validation set rather than to the training set because of information leakage during the hyperparameter fine-tuning and model selection process. Therefore, reporting the performance of

the final model on a separate (unseen) test set, and ideally on external validation datasets if applicable, is crucial for verifying the model's generalizability.

Table 2. 3: A lists of common methods to mitigate overfitting

How to mitigate Overfitting
More training data
Data Augmentation
Regularization (Weight decay, dropout)
Batch Normalization
Reduce Architecture Complexity

2.17. Related Works

Automated currency recognition and classification system have attracted tremendous research as the need for currency recognition in banks, ATMs, vending machines, etc. Several designs or techniques have popped up over time. So in this chapter, some related work to our topic is summarized and discussed.

2.17.1. Related works on Ethiopian paper currency

Zeggeye et.al. [13] described their design of automatic recognition of Ethiopian banknotes, where hardware and software solutions worked on taking images of an Ethiopian banknote from scanner and camera as an input. Four characteristic features, i.e. the dominant color, the distribution of the dominant color, the hue value, and speed up robust features (SURF) were extracted as the discriminative features of the banknotes. Those features in combination with the local features descriptors were involved in a four-level classification process, a classification task executed every time one of the four features was extracted. The correlation coefficient based template matching was implemented for classification. Test result showed that the proposed design had an average recognition rate of 90.42% for genuine Ethiopian currency, with an average processing time of 1.68 seconds per banknote. Here, even though the authors have used a suitable method for classification and counterfeit detection, the methods suffers accuracy problem.

Additionally, Tessfaw et. al. [14] proposed a system called "Ethiopian Banknote Recognition and Fake note Detection Using support Vector Machine ". The system basically contains basic image processing techniques such as image acquisition, image preprocessing, feature extraction and classification using support vector machine. Basically camera or scanner used for image acquisition. The images of currency processed using a variety of preprocessing techniques and different features of the image extracted using local binary pattern technique, once the features are extracted it is important to recognize the currency using effective classifier called Support vector machine and Finally a prototype able to recognize Ethiopian paper currency with accuracy of 98% shows high performance classification model for paper currency recognition and also verify the validity of given banknotes with average accuracy of 93% rate. However, the method or classifier used here has high algorithm complexity and run slowly.

2.18. Summary

Table 2. 4:: Related work summary

Ref No.	Title	Method	Accuracy	Gaps
[9]	Indian Banknote Recognition using CNN	Transfer Learning (Fine Tuning) on pre-trained Mobile net	96.6%	The experiment was done only on a single dataset split ratio. (0.4:0.2:0.2)
[13]	Automatic Recognition of Ethiopian Paper Currency	Combined the approaches of currency characteristic comparison and local feature descriptors	90.42%	Preprocessing techniques and Classification algorithm used.
[10]	Currency detection and recognition on deep learning	SSD model based on DL as the framework and CNN model to extract the feature of the banknote	96.6%	The classification part can be improved by using other methods.
[14]	Ethiopian Banknote Recognition and	Basic image processing techniques for feature extraction and SVM for	98%	Banknote feature extraction work can be improved by using advanced image

	Fake Detection using SVM	classification		processing techniques.
[15]	Ethiopian paper Currency Recognition System	Color momentum, SIFT, GLCM, and CNN as feature extraction techniques and FFANN as a classifier	99.4%	The classification method of this work can be improved with advanced architecture like GoogLeNet, MobileNet, and ResNet using a larger dataset.

CHAPTER THREE: RESEARCH METHODOLOGY

The purpose of this chapter is to provide a detailed description of the research method used in this thesis and the "Materials" employed while doing it. Section 3.1 describes the general research description. Section 3.2 describes the Data collection. Section 3.3 describes Data Preprocessing. Section 3.4 describes the Data augmentation. Section 3.5 describes Convolutional Neural Networks (CNNs). Section 3.6 describes MobileNet. Section 3.7 describes the rationale and choice of Technology. Section 3.8 describes CNN Training. Section 3.9 describes Transfer Learning, and Section 3.10 describes software tools used.

3.1. Research Design Description

The block diagram shown in figure 3.1 below describes the basic concept of this specific research process or flow. It is a methodological analysis for the proposed model. It starts from banknotes images dataset preparation as a prerequisite and includes image processing, different software tools, pre-trained mobile Net model selection, installation, training the CNN, and finally evaluating the models based on the test dataset. The model which has less loss value and highest accuracy was considered as a model proposed in this study.

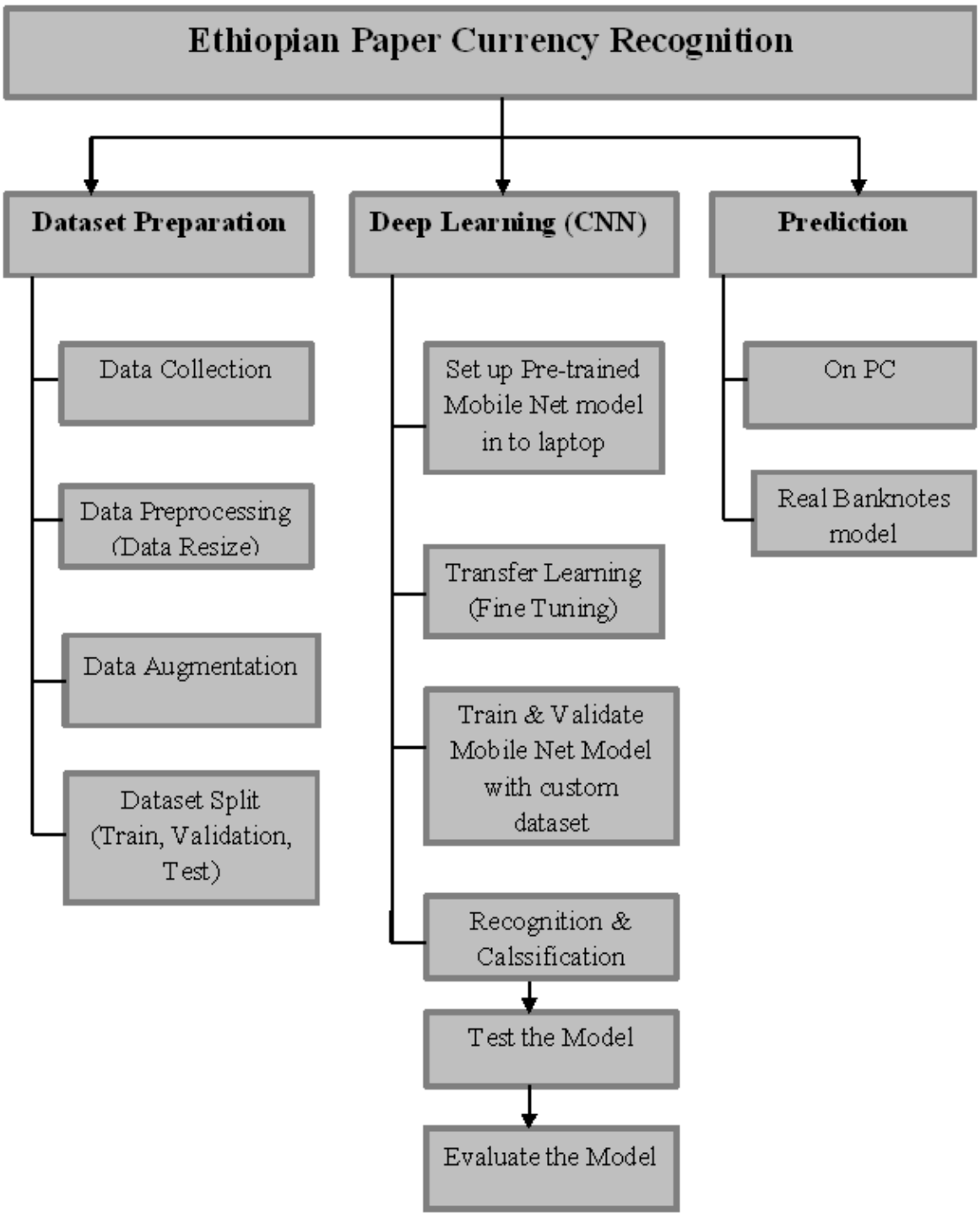


Figure 3.1:Block Diagram of the Model

3.2. Data Collection

In this research, four newly released Ethiopian paper currency with denominations 10ETB, 50ETB, 100ETB, and 200ETB were collected from the Commercial Bank of Ethiopia (CBE). These collected banknotes' physical conditions are very new, new, normal, circulated, and folded. The image was captured from one side of the banknote with various backgrounds by using two smartphone cameras and a scanner with different resolutions. The descriptions of these devices are given in Table 3. 1 below. In the case of capturing banknote images, the various lighting conditions namely, indoor lighting, dim light, artificial light, as well as daylight attempt were included. And also all banknote images were taken in uncontrolled environments. So different protrusions like shadows and fingers are included. Here manual labeling is done using Adobe Photoshop and 110 images of each banknote are selected so that a total of 440 original images are selected as a dataset.

3.2.1. Data Sources

To obtain the required image data, the primary data is acquired from four denominations of newly released Ethiopian paper currency. All banknotes are very new, new, normal, folded, and to some extent circulated. These currencies were collected from the Commercial Bank of Ethiopia. Images of these banknotes are taken using different phone cameras and a scanner with different resolutions. Since the primary data source is too small in size, data augmentation techniques were used to increase the dataset artificially.

3.1.1.1. Primary Data Sources

Having a high-quality dataset is a mandatory condition to be filled to have a model performing efficiently. So the performance of the model is highly dependent on the quality of image datasets that are used to train a model. Here different cameras with high resolutions were applied to acquire banknotes image. The primary data from which the datasets were prepared were collected from newly released Ethiopian banknotes. All these currencies were obtained from the Commercial Bank of Ethiopia. In general, as the primary data sources

newly released Ethiopian banknotes with four denominations (10ETB, 50ETB, 100ETB, and 200ETB) were used in this study as shown in figure 3.2.

Table 3. 1: Number of data collection for All ETB denominations

No	Denominations	Camera	Scanner	Total Pictures
1	10 ETB	2	1	110
2	50 ETB	2	1	110
3	100 ETB	2	1	110
4	200 ETB	2	1	110
Total				**440**

Figure 3. 1:The four newly released Ethiopian banknote denominations

3.3. Data Pre-Processing to resize images

As most of the neural network models assume a square shape input image, and our datasets are arbitrarily sized RGB images; it is reshaped or resized to 224 X 224 pixels maintaining a uniform aspect ratio. There is no standard set to resize our images but we can resize them based on the computational resources we have because as the size of the image increased, it needs more computational resources to be processed. In this study, banknote images were reshaped or resized to 224 x 224 pixels to make them suitable for feeding to the pre-trained MobileNets model used.

3.4. Data Augmentation for Increasing Dataset size

When we have limited training data in deep learning, overfitting will occur, and Data Augmentation is the best way to combat this. So that we introduced the technique of image augmentation for artificially growing datasets because the neural network appears to over-fit in the case of a small number of training data samples trained with a higher number of epochs. In this work, the images have been augmented by subjecting them to rotations, horizontal, shifts, width shifts, height shifts, shear, horizontal flip, and zoom. Table 3.2 shows details of the features used to augment the dataset. Each original banknote image has been subjected to such features 14 times, followed by shuffling of images, resulting in a 14 times larger shuffled dataset with a total of 6600 images. For each denomination, we have got 1540 augmented banknote images and 110 original images. Therefore, each denomination has a total of 1650 images. Figure 4.1 in the next chapter shows sample images from the augmented dataset.

Table 3. 2: Details Datasets Augmentation Techniques

S. No.	Augmentation Type	Range
1	Rotation	0 to 45 degrees
2	Horizontal Shift	0 to 10 %
3	Brightness	± 10 %
4	Vertical Shift (for camera images only)	0 to 10 %
5	Zoom in/out	± 10 %
6	Horizontal Flip	

3.5. Approaches of Banknotes Images Processing

3.5.1. Deep Convolutional Neural Network (CNN)

Deep convolutional neural networks are machine learning models that can learn high-level features from the low-level in a hierarchical fashion, which is translated by the stacking of operational neural network layers, essentially convolutional layers, one above the other. The convolutional architecture of DCNNs is inspired by the visual cortex of the animal brain,

where neurons are arranged in a way that they produce a response to the overlapping regions in their visual field [35]. The CNN was originally applied to document recognition [36], however, it finds wide applications in several domains, where the data can be represented in two or more dimensional matrices and the arrangement of data elements in such matrices carries significance. Hence, CNN's are as well applicable to areas of video recognition, natural language processing, etc. Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects, and traffic signs apart from powering vision in robots and self-driving cars [ ].

3.5.1.1. CNN Architecture

Given a large set of images and their classification labels, CNN can learn a hypothesis to classify new images into their respective classes. This is done by transforming the image data to the class predictions by passing it through a set of operations dictated by the constituent layers of the network. Figure 3.4 shows such operations on a single channel (grayscale) image till the class prediction scores, in a typical CNN. The idea is easily extensible to 3- channels (RGB, L*a*b, etc.) images, as is the case used in this study. The four key operations of the CNN Architecture are:

A. Convolutional Layers
B. Non-Linearity (ReLU)
C. Pooling Layers
D. Fully Connected Layers (FC)

Since these operations are the fundamental building blocks of a Convolutional Neural Network, knowing how they function is critical to gaining a thorough understanding of ConvNets.The intuition behind each of these operations is discussed below.

A. Convolutional Layer

The primary purpose of Convolution in the case of a ConvNet is to extract features from the

input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. It consists of a set of filters, which are convolved with the image to detect the presence of features dictated by them. Each filter is convolved with an image to produce a feature map, which is an array containing convolved output at each pixel. So, as shown in figure 3.3 below, each of the 3×3 filters contributes to a feature map of size 32×32, if the zero paddings are applied before convolving. Each feature map will have positive response values at positions where the feature was found, while negative values or weak responses at remaining positions.

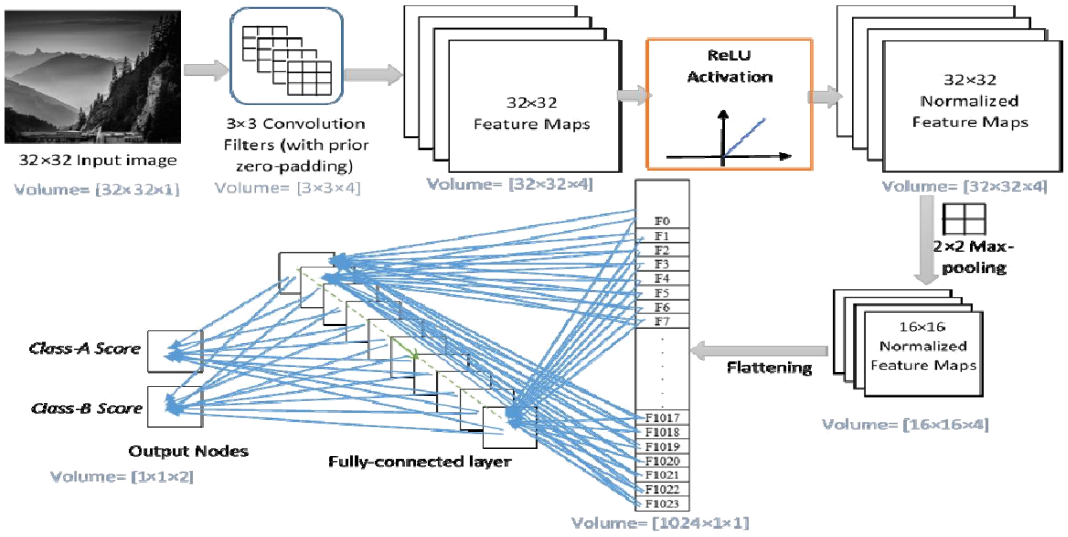


Figure 3. 2: Illustration of the operations involved in a typical CNN during image classification (Taken from [9]).

B. Non- Linearity (RELU)

It aims at normalizing the feature maps by changing every negative valued element to zero and leaving the rest unchanged. ReLU activation function is mathematically defined as,

g(x) = max(0,x)……..(3.1)

where g(x) is output and x is the value of feature element or input

Equation 3. 1: Equation of RELU function

C. Pooling or Sub Sampling Layer

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum, etc. In the case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element, we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

The figure below shows an example of a Max Pooling operation on a Rectified Feature map (obtained after convolution + ReLU operation) by using a 2×2 window.

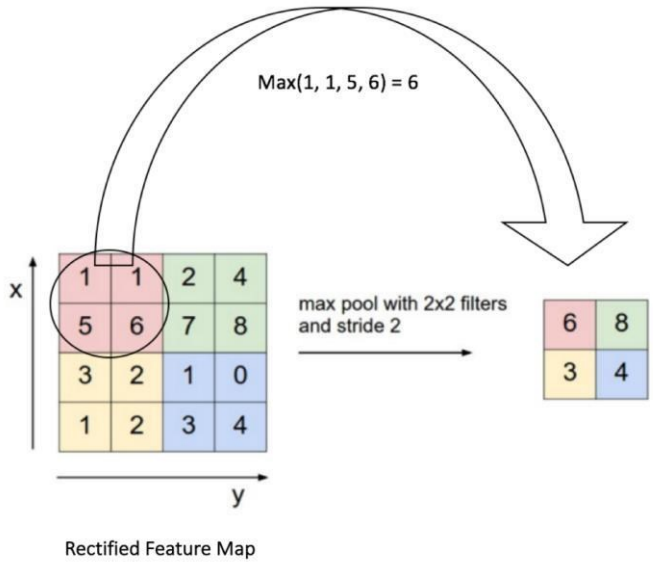


Figure 3. 3: Pooling Layer

We slide our 2 x 2 window by 2 cells (also called 'stride') and take the maximum value in each region. As shown in the figure above, this reduces the dimensionality of our feature map.

D. Fully Connected Layer

The Fully Connected layer is a traditional Multi-Layer Perceptron that uses a softmax

activation function in the output layer in this study. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represents high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. For example, the image classification task we set out to perform has four possible outputs.

3.6. Train Using Backpropagation

During filtering of the image right up to the output scores, the network needs to decide the optimal values of filter weights as well as voting weights for the correct classification. This is done through the process of backpropagation, where the prediction scores are compared with the actual classification answers to find the error. Weights are then adjusted depending upon this error by propagating back through the whole network. These adjustments are made using the technique of gradient descent.

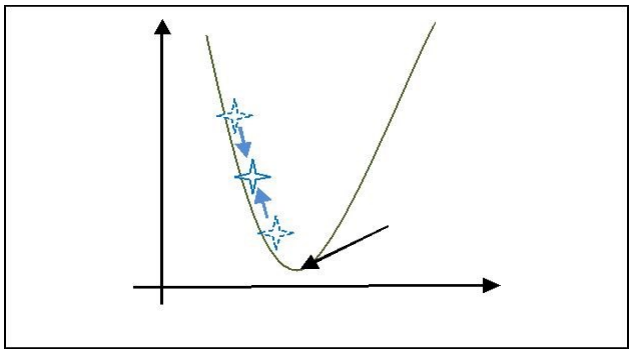


Figure 3. 4: Gradient Descent Optimization

As shown in Figure.3.5, for each feature pixel and voting weight, its value is adjusted up and down and the change in error is observed. So, the optimization problem boils down to reach the valley where the error is least, through adjustment of weights. This can be accomplished in several iterations of the backpropagation during training.

3.7. Optimization Algorithms

The loss function of a convolutional neural network is highly non-convex. so that gradient-based optimization algorithms can be applied. However, convolutional neural networks are usually made of millions of parameters. Thus, only the first-order derivatives are used in practice the most widely used first-order optimization algorithm is Gradient descent. These algorithms tell us whether the function is decreasing or increasing at a particular point it gives a tangential line to a point on its error surface as shown in figure 3. 5 above.

3.8. MobileNets

MobileNets [37] refers to a class of specially designed convolutional neural networks with an aim of resource friendliness and model scalability to suit the needs of mobile platforms. These come with two simple hyperparameters that the designer can easily find to suit the resource-vs-accuracy trade-off as per the problem at hand. The width multiplier facilitates the adjustment of network width, whereas the resolution multiplier enables to scale of the input image dimensions, by adjustment of internal layer representations.

MobileNets differ from conventional convolutional neural networks in that- they use 3×3 depth-wise convolutions and 1×1 point-wise convolutions instead of 3×3 convolutions followed by batch normalization and rectified linear units (ReLUs). Google open-sourced the models of MobileNets for different configurations, which come pre-trained upon ImageNet [38] database. The pre-trained model configurations vary from the smallest one with - width multiplier of 0.25 and an image resolution of 128×128, and the largest one with – with a multiplier of 1.0 and an image resolution of 224×224 pixels. Since the proposed work involved the training and testing of the classification framework upon an indigenous personal computer (PC), there was not much constraint upon the computational resources. Hence, the work used the largest version of MobileNet – width multiplier equal to 1.0 and image resolution of 224×224 pixels.

3.9. Transfer Learning

Training deep learning models from scratch requires a huge amount of training data so that the network parameters converge to an optimal solution that generalizes the problem at hand. With a smaller dataset, on the other hand, the network tends to over-fit to the given instances. Apart from the need for large datasets, learning a deep network requires tuning of learning hyper-parameters like the choice of the gradient optimization algorithm, learning rate, number of iterations of learning, etc. Such hindrances can be fairly avoided using the strategy of transfer learning.

The concept of transfer learning aims to reuse the existing pre-trained network- the network whose weights (parameters) have gained the maturity to reach convergence on a large dataset involved in a problem. With the problem in hand, the approach of transfer learning can be decided. Most often the classification problem can be looked upon considering two important aspects – the size of the dataset, and its similarity with the dataset upon which the network comes pre-trained. In this work, the softmax layer of the pre-trained model is replaced by a new one and trained afresh from the bottleneck features extracted from banknotes images.

3.10. Performance Measurement Methods

3.10.1 Loss Function (objective function)

The loss function is the crucial element[39] in artificial neural networks, which is used to measure the inconsistency between the predicted value (y^) and the actual label (y). It is a nonnegative value, where the robustness of the model increases along with the decrease of the value of loss function. The loss function is the hardcore of empirical risk function as well as a significant component of the structural risk function.

3.11. Evaluation Metrics to Evaluate the Accuracy of the Model

In this research work, various performance evaluation matrixes are used to evaluate the trained model, including simple accuracy, loss, and confusion matrix.

3.11.1. Accuracy

The classification accuracy is determined by the total number of correctly classified currencies divided by the total number of validation as shown in Equation 3.2)

$$\text{Accuracy} = \frac{\text{Correctly Classified Currency}}{\text{Total Number of Validation}} \times 100 \quad \text{.....(3.2)}$$

Equation 3. 2: Accuracy equation

3.11.2. Confusion Matrix

A more detailed wrong and correct classification for each Ethiopian currency class are shown by using the confusion matrix because it can illustrate the correct and incorrect classification percentage for each class in a tabular way.

3.12. Data Analysis

The data that we got finally is represented in terms of different ways. The performance of the different models is represented in terms of percentage (%) and the different hypermeter and parameter values we tuned up will be analyzed using scalar integer value for each parameter and the performance of the cost function of the final model will be shown using graphs, and finally the tuned hypermeters and parameters for each model.

3.13. Software Tools

As software tools to develop a CNN-based model in this thesis, different machine learning libraries (TensorFlow, Keras), data science libraries (Pandas, NumPy), visualization libraries(Matplotlib), Python programming language, and Jupyter notebook were used. Also for data preprocessing (image resize), data augmentation, and data splitting, some small programs were written and applied. The details of these basic software tools applied in this study are depicted below.

3.13.1.Programming Language

3.13.1.1. Anaconda Distribution

Anaconda is a **distribution** of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. It is strongly recommended as it offered coding for popular python frameworks like Sckikit, NumPy, Scipy, panda, etc. It also consists of a popular platform like Jupyter notebook, is reduces the importing time for libraries, pre-installed many important libraries for data science. It has many preinstalled packages that will be used in data science like NumPy, pandas, Scipy, etc. More than 1500 puls packages can easily be installed with it. It is one of the most popular python distributions in the world with over 11millions users.

3.13.1.2. Python

The entire work of the Thesis is coded in Python programming language. Python is a widely used high-level, easy, powerful, general-purpose, interpreted, dynamic programming language [40]. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in other high-level languages. Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python allows the simple representation of large data matrix and vectors and also supports external libraries as required. Due to this feature of python, we were able to implement the algorithm successfully with minimum lines of code. Thus, the major reasons behind the use of python programming language for this thesis are Software quality, Developer productivity, Program Portability, Support Libraries, and Component Integration to mention a few.

3.13.1.3. Jupyter Notebook

Is an open-source web application, a widely used application in data science, for creating and sharing documentation that contains code, visualizations, and documentation. It can be

integrated with many programming languages but it is widely used for python. It is a very fast web-based application to execute code.

3.13.1.4. Numpy

NumPy is the high-performance numeric programming extension for python. It is the core library for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms basic linear algebra, basic statistical operations, random simulation and much more.

3.13.1.5. Pandas

Pandas is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python has long been great for data munging and preparation, but less so for data analysis and modeling. Pandas help fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain-specific language like R.

3.13.1.6. Matplotlib

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named Pyplot which makes things easy for plotting by providing a feature to control line styles, font properties, formatting axes, etc. It supports a very wide variety of graphs and plots namely - histograms, bar charts, power spectra, error charts, etc. It is used along with NumPy to provide an environment that is an effective open-source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython [41].

3.13.1.7. TensorFlow

Tensor Flow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models and it‘s an open-source library for fast numerical computing, which is an end-to-end [37] open-source machine

learning(ML) framework. Tensor Flow was designed for use both in research and development and in production systems. It can run on single CPU systems, GPUs as well as mobile devices, and large-scale distributed systems on hundreds of machines. So we can say TensorFlow is a brilliant tool, with lots of power and flexibility. However, for quick prototyping work, it can be a bit verbose.

3.13.1.8. Keras

Keras is a high-level neural network, API[42], written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. It is a model-level library providing high-level building blocks for developing deep learning architectures. It does not allow low-level operations like tensor manipulation and differentiation, Instead of that, it uses a specialized, well-optimized tensor library, which serves as a backend engine for Keras. Keras allows for easy and fast prototyping through user-friendliness, modularity, and extensibility.

3.13.1.9. Web Browser

To open the Jupyter notebook application we need a web browser the browser uses the localhost port to run on the local machine.

CHAPTER FOUR: IMPLEMENTATION

4.1. Overview

This Chapter provides the implementation details of the model and data preparation. The implemented model is done using Tensor flow's Keras API framework. The model was selected and implemented using fine-tuning. Mobile Nets are built on depthwise separable convolution layers. Each depthwise separable convolution layer consists of a depthwise convolution and a pointwise convolution. Counting depth-wise and pointwise convolutions as separate layers, a Mobile Net has 28 layers. These convolutions are discussed below. Finally, there is a SoftMax classifier with four classes that categorizes the four Ethiopian banknotes into their respective denominations.

4.2. Environment Setup

Train CNNs from zero (scratch) requires a lot of data and high-performance computing powered hardware. The training and evaluation of the model are implemented using a Tensor flow's Keras API which is configured in Windows 10 environment. All the training, preprocessing, and experimental tasks were done on Toshiba 450 G3 laptop with Intel(R) Core(TM) i7-5500U CPU @ 2.40GHZ processor and 8 GB RAM having Windows 10 operating system. All the codes presented in this study are written by using Python programming languages version 3.8.5. The idea behind using python as a programming language is its capabilities of ease to learn and the availability of matured resources to perform computer vision and real-time techniques [17]. Tensor flow is a completely open-source framework developed by Google in 2015 by holding the ambition of being a playing place for machine learning. This framework is written in C++, Python, and Cuda. There are so many reasons behind the implementation of this framework some of them are speed for computation which makes Tensor flow appropriate for the practical industry and academic research, expressive architecture, matured online support, and resource availability [18].

4.3. Experimental setup

Transfer learning is applied by using the pre-trained models Mobile Net to train the prepared newly released Ethiopian currency custom dataset. As mentioned in the framework selection section, Tensor flow's Keras API was used to train the dataset. A standard Mobile Net has 4.2 million parameters which can be further reduced by tuning the width multiplier hyperparameter appropriately. The size of the input image is 224 × 224 × 3. After Fine-tuning, the parameters of this standard Mobile net model is reduced to 3.2 million parameters (See Appendix D).

4.4. Dataset Preparation

In the data preparation task image labeling using Adobe Photoshop 2015 and data separation was performed. Image labeling is an essential task for the supervised machine learning techniques because the output result of the model is determined by the labels we feed the model in the training stage. For each image, the labeled information was saved as jpg format compatible for CNN pre-trained models. We attempt to be rational in the case of include or discard banknotes from labeling. Banknotes that were fully or partially visible and recognizable were included, whereas banknotes that were unrecognizable because of size or position were excluded. The images presented in the dataset have similar width and height which is 256×256 then resizes the images to the target size which is 224 × 224 × 3 resolution. Here before we start training the model, we need to know about the size of the datasets we have and the computational power going to be used. Even if we cannot afford the best computational power we want, at least we need to have a sufficient dataset to train our model. Therefore before starting to train the model, it is needed to well prepare the data. Originally collected images from each newly released Ethiopian paper currency denominations are 110 images. These datasets are not enough to use for our deep learning model. So the first thing we need to do is to increase the size of these data set by using Data Augmentation. Thus 440 total original images are augmented into 6160 images. From these prepared images 1540 images is collected from 10ETB, 50ETB, 100ETB, and 200ETB

denominations. After the completion of data augmentation, splitting the dataset into training, validation, and test dataset which is used for train the model and is used for evaluating the trained model respectively was performed before the actual training begins. The data splitting task was done by adopting the ratio of 08:01:01(80% training, 10% validation, and 10% testing) and 07:15:15(70% training, 15% validation, and 15% testing). This means that 80% of the dataset which is 4828 images are used to train the model, 10% from the dataset which is 616 images are used to validate the model, and 10% from the dataset which is 616 images are used to test the trained model. Also 70% from the dataset which is 431 images are used to train the model, 10% from the dataset which is 924 images are used to validate the model, and 10% from the dataset which is 924 images are used to test the trained model. These two types of dataset categories shown in Table 5.1 and Table 5.2 below are used for comparison of the model.

Table 4. 1: Data Splitting percentages and purpose

Datasets	**Training**	**Validation**	**Testing**
% of total dataset	80%	10%	10%
Number of images	4928	616	616

Table 4. 2: Data Splitting percentages and purpose

Datasets	**Training**	**Validation**	**Testing**
% of total dataset	70%	15%	15%
Number of images	4312	924	924

4.5. Developing the Model

The developed model is a standard Mobile Net model which is a model with 28 layers. The model is programmed using python programming on a Tensor flow's Keras framework by using the convolutional neural network. The implementation of the model is shown in appendix C. Then the model will be trained using the training dataset The implementation is done on appendix C and then the model hypermeters will be tuned (tried on different values)

using the development dataset as a result different models will be generated and the model with the best performance will be selected and tested using the test dataset.

4.6. Mobile Net Model Architecture

Mobile Net [12] was introduced by researchers at Google in 2017. The network is different from the traditional convolution network. The network is small and fast. The unique aspect of Mobile Net which differentiates it from other CNNs is the use of depth-wise separable convolution split which means that it splits the convolution into a 3x3 depth-wise convolution and a 1x1 point-wise convolution. Counting depth-wise and pointwise convolutions as separate layers, a Mobile Net has 28 layers. A standard Mobile Net has 4.2 million parameters which can be further reduced by tuning the width multiplier hyperparameter appropriately. The size of the input image is 224 × 224 × 3. The architecture of Mobile Net is shown in Table 4.3: below.

Table 4. 3: Mobile Net Body architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

It has been known in the machine learning community that fully trains of CNN network to a task-specific dataset needs huge computer resources and take time. Most of all, the dataset size must affect performance. For the reasons, the transfer of learning approach has been come up and many of the well-known CNN pre-trained architectures are public. In this study, pre-trained architectures trained on the based task of the Image Net dataset are explored. We select standard pre-trained mobile net architecture with 4.2 million parameters and 224x 224 input sizes shown in Table 4.3 above. Such architectures are public as the pre-trained networks with natural images in the Image Net dataset. By using the CNN training method called "Fine Tuning" we reduced the parameters to 3.2 million and develop the model. The architecture of the developed model is shown in appendix D.

After the model has been developed and the dataset is ready to be feed to the model there are some parameters that we need to train the model. We need to specify parameters such as Batch size, image size, and the number of epoch, learning rate, and optimizer. An optimizer is an algorithm that will be used to make gradient descent so that to enable the network to learn. The optimizer that we are going to use is the Adam optimizer. We chose the Adam optimizer because it has the combined befits of both AdaGrad and RMSProp. It has also a lot of advantages such as

- The method is straightforward to implement.
- Is computationally efficient.
- Has a little memory requirement.
- Is invariant to a diagonal rescaling of the gradient.
- Suitable for problems with large data and parameters.
- Suitable for noisy and/or sparse gradients.

CHAPTER FIVE: RESULT AND DISCUSSION

5.1. Result

After performing all the evaluations on two types of split datasets ratio on the developed model, experiments with top results are discussed below. The model is evaluated using different parameters that can affect the efficiency of the model. These parameters are the number of epochs; train test split ratio, and learning rate which are explained briefly in Chapter 4. Different parameter ratios are used for the evaluation purpose and the most successful ratios are discussed in this chapter.

Note: When we read our confusion matrix, it is classified into two labels in the table, which are the horizontal line is labeled as predicted value whereas the vertical line is labeled as the True Value.

5.1. Table Summary for Evaluation of the Model

Table 5. 1: Summary for Evaluation of the Model

No	*Dataset type*	*Epochs*	*Learning Rate*	*Test Ratio*	*Accuracy*	*Validation*	**Loss**
1	80/10/10 Dataset Ratio	3.0	Adam(0.01)	10%	0.9915	0.9221	0.0319
		3.0	Adam(0.001)	10%	0.9972	0.9886	0.0093
		3.0	Adam(0.0001)	10%	0.9998	1.0000	0.0061
		5.0	Adam(0.0001)	10%	1.0000	1.0000	$8.5011e^{-04}$
2	70/15/15 Dataset Ratio	3.0	Adam(0.01)	15%	0.9861	0.9708	0.0643
		3.0	Adam(0.001)	15%	0.9949	0.9903	0.0133
		3.0	Adam(0.0001)	15%	1.0000	0.9989	0.0061
		5.0	Adam(0.0001)	15%	1.0000	1.0000	$9.4565e^{-04}$

The above table summarizes the top results found for the evaluation of the model on two types of datasets split ratio which is used for comparison. These dataset type ratios are the

80/10/10 dataset ratio and 70/15/15 dataset ratio. The performances of those models are measured in terms of the loss value they have and the accuracy value they have achieved.

5.2. Evaluating the Model on 80/10/10 dataset ratio.

Table 5. 2: Result summary of the model on 80/10/10 dataset ratio

No	*Epochs*	*Learning Rate*	*Test Ratio*	*Accuracy*	*Validation*	**Loss**
1	3.0	Adam(0.01)	10%	99.15%	92.21**%**	03.19**%**
2	3.0	Adam(0.001)	10%	99.72%	98.86**%**	0.93**%**
3	3.0	Adam(0.0001)	10%	99.98%	100%	0.61**%**
4	**5.0**	**Adam(0.0001)**	**10%**	**100%**	**100%**	**8.5011e^{-04}**

The above table summarizes the four results found when evaluating the model on 80/10/10 dataset ratio using different learning parameters and their effects on the model. As we can learn from the table every parameter change has its effect on the result the model is delivering with the training time needed and accuracy of the model. Let‘s discuss some results that are affected by the change made on different parameters.

5.2.1. Result comparison of the model on 80/10/10 dataset using Learning Rate

From Table 5. **2**, we can take results on rows 1,2 and 3, which have the same parameters except they differ in their learning rates, in which result on row 1 has LR of 0.01, result on row 2 with LR of 0.001, and result on row 3 with LR of 0.0001. By comparing the three results concerning their learning rate we can conclude the effect of the learning rate on the model as follows:

- When the LR increases from 0.01 to 0.0001 with the number of epoch = 3, training accuracy and validation are increased.

From Table 5. **2**, we can take results on rows 3 and 4, which have the same parameters except they differ in their number of the epoch. The number of epoch for row 3 is 3 and the number of the epoch is for row 4 is 5. By comparing the two results concerning their learning rate we can conclude the effect of the learning rate on the model as follows:

- Having constant values of LR, with an increased number of epochs the values of training accuracy and validation are increased. As the number of epochs increases in the model or sample dataset, the accuracy of the model increases. In the following figure 5.1, 5.2, 5.3, and 5.4 train model, accuracy, loss, and confusion matrix can be used for visualization.

Train the Model

80/10/10 Dataset Split Ratio when epoch = 5, Adam (0.0001), Batch Size = 32

```
Epoch 1/5
154/154 - 557s - loss: 0.0033 - accuracy: 0.9996 - val_loss: 0.0028 - val_accuracy: 1.0000
Epoch 2/5
154/154 - 425s - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0023 - val_accuracy: 1.0000
Epoch 3/5
154/154 - 416s - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0026 - val_accuracy: 1.0000
Epoch 4/5
154/154 - 414s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 5/5
154/154 - 416s - loss: 8.5011e-04 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
```

Figure 5. 1: Training performance 80/10/10 Dataset Split Ratio

Accuracy Graph

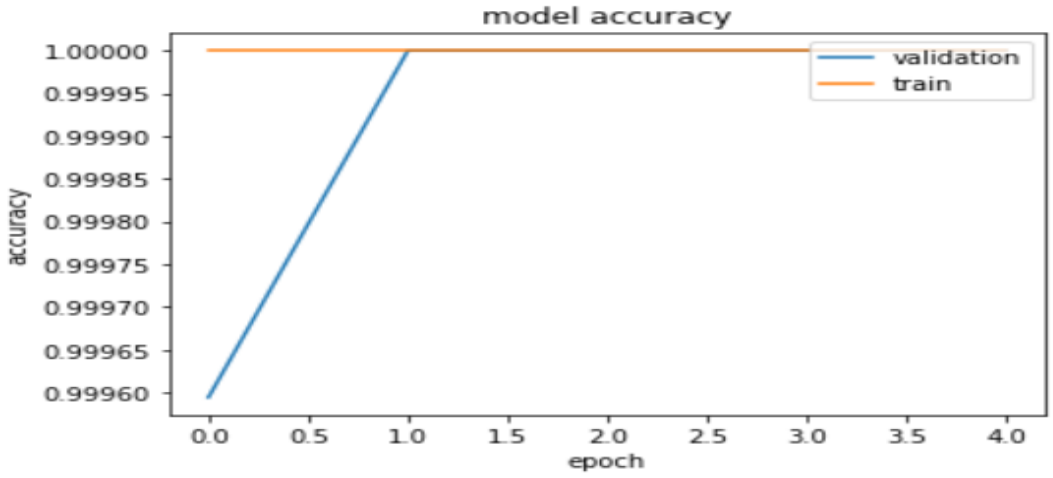


Figure 5. 2: Accuracy Graph of Train and Validation on 80/10/10 ratio

Loss Graph

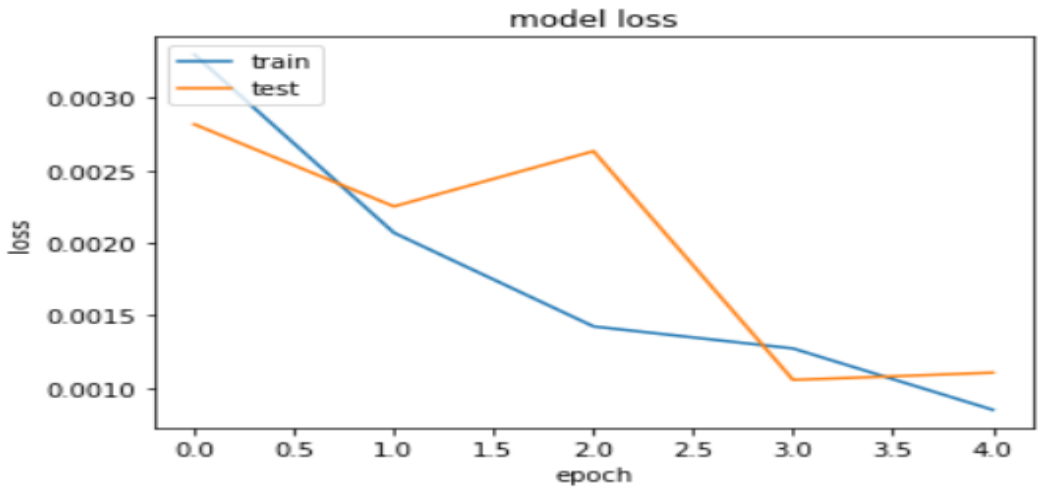


Figure 5. 3: Loss Graph of Train and validation on 80/10/10 ratio

Confusion Matrix

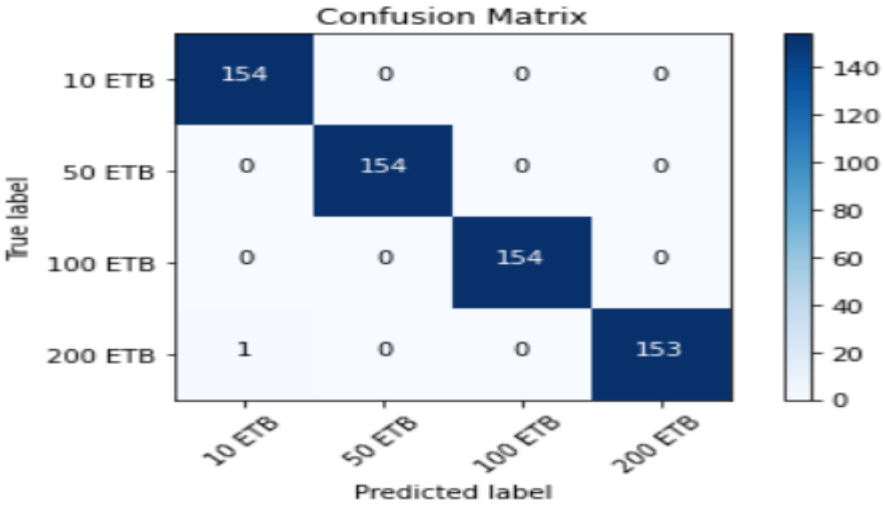


Figure 5. 4: Confusion matrix of result for Table 5

5.3. Evaluating the Model on 70/15/15 dataset ratio.

Table 5. 3: Result summary of the model on 70/15/15 dataset ratio

No	*Epochs*	*Learning Rate*	*Test Ratio*	*Accuracy*	*Validation*	*Test*	Loss
1	3.0	Adam(0.01)	15%	0.9861	0.9708		0.0643
2	3.0	Adam(0.001)	15%	0.9949	0.9903		0.0133
3	3.0	Adam(0.0001)	15%	1.0000	0.9989		0.0061
4	**5.0**	**Adam(0.0001)**	**15%**	**1.0000**	**1.0000**		**$9.4565e^{-04}$**

The above table summarizes the four results found when evaluating the model on 70/15/15 dataset ratio using different learning parameters and their effects on the model. As we can learn from the table every parameter change has its effect on the result the model is delivering with the training time needed and accuracy of the model. Let‘s discuss some results that are affected by the change made on different parameters.

5.3.1. Result comparison of the model on 70/15/15 dataset using Learning Rate

From Table 5. **3**, we can take results on rows 1,2 and 3, which have the same parameters except they differ in their learning rates, in which result on row 1 has LR of 0.01, result on row 2 with LR of 0.001, and result on row 3 with LR of 0.0001. By comparing the three results concerning their learning rate we can conclude the effect of the learning rate on the model as follows:

- When the LR increases from 0.01 to 0.0001 with the number of epoch =3, training accuracy and validation are increased.

From Table 5. **3**, we can take results on rows 3 and 4, which have the same parameters except they differ in their number of the epoch. The number of epoch for row 3 is 3 and the number of the epoch is for row 4 is 5. By comparing the two results with respect to their

learning rate we can conclude the effect of the learning rate on the model as follows:

- Having constant values of LR, with an increased number of epochs the values of training accuracy and validation are increased. As the number of epochs increases in the model or sample dataset, the accuracy of the model increases. The following figure 5.5, 5.6, 5.7, and 5.8 train model, accuracy, loss, and confusion matrix can be used for visualization

Train the Model

70/15/15 Dataset Split Ratio when epoch = 5, Adam (0.0001), Batch Size = 32

```
Epoch 1/5
135/135 - 380s - loss: 0.0041 - accuracy: 0.9998 - val_loss: 0.0053 - val_accuracy: 0.9989
Epoch 2/5
135/135 - 422s - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.0039 - val_accuracy: 0.9989
Epoch 3/5
135/135 - 421s - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.0038 - val_accuracy: 0.9989
Epoch 4/5
135/135 - 417s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0030 - val_accuracy: 0.9989
Epoch 5/5
135/135 - 390s - loss: 9.4565e-04 - accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 1.0000
```

Figure 5. 5: Training performance 70/15/15 Dataset Split Ratio

Accuracy Graph

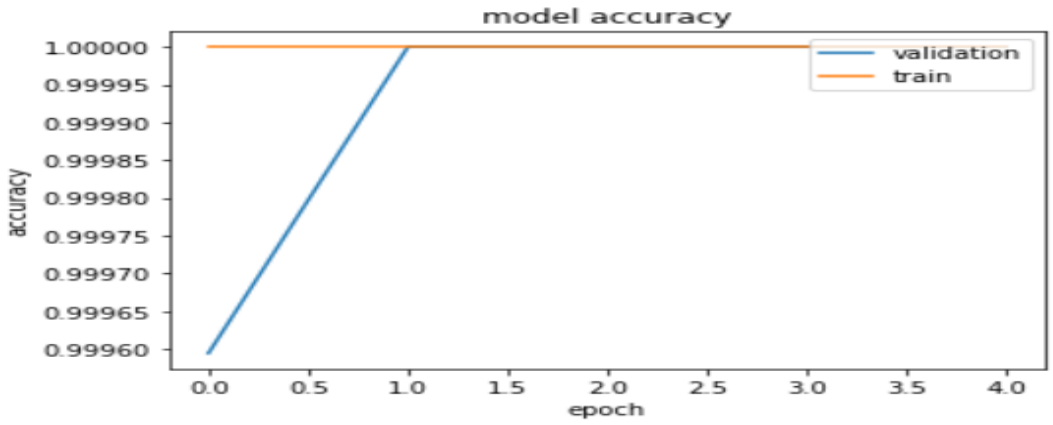


Figure 5. 6: Accuracy Graph of Train and Validation on 70/15/15 ratio

Loss Graph

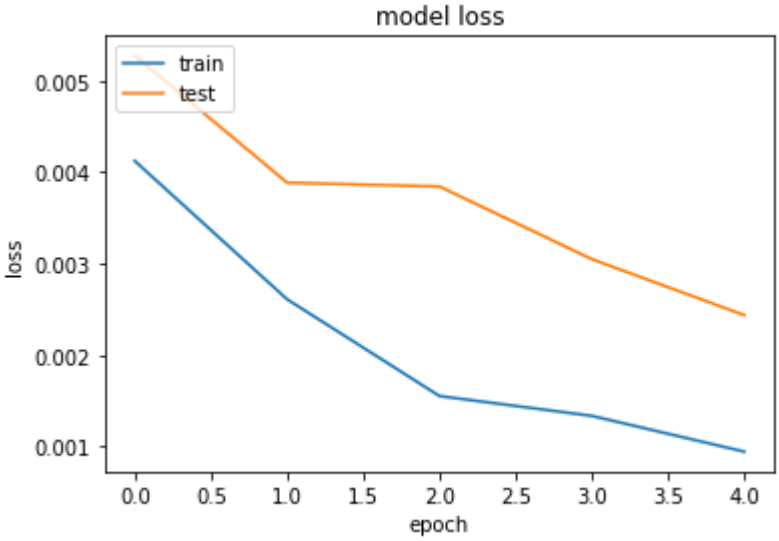


Figure 5. 7: Loss Graph of Train and validation on 70/15/15 ratio

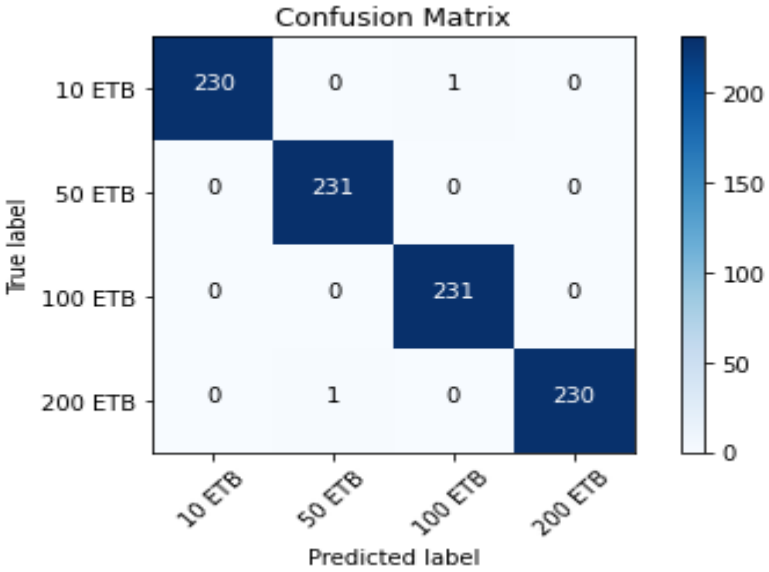


Figure 5. 8: Confusion matrix of result for Table 5.3.

As we see from the Table5.2, and Table 5.3 above, on average when we use the 80/10/10 proportion of the training and we obtained high performance on the loss value, training accuracy value, and validation accuracy value, so an increase in the training dataset increased

the performance of the model.

On the other side, the 70/15/15 has made the training smaller and the test dataset high compared to the 80/10/10 so the performance we obtained on the 70/15/15 is smaller on the loss value, on the training accuracy, and the validation accuracy than the 80/10/10 category. So by increasing the training dataset as much as possible we have increased the overall performance of the training performance as well as the test performance for all the models.

CHAPTER SIX: CONCLUSION AND RECOMMENDATION

6.1. Conclusion

In this study, the researcher attempted to develop and tested models that provide a capability to recognize newly-released Ethiopian paper currency by using transfer learning. The model can classify Ethiopian currencies into their respective categories. In-depth reviews of currency recognition studies are performed. Pre-trained MobileNet models are used and also trained by using a custom dataset and evaluated.

Even if there are few pieces of research conducted in the area of Ethiopian paper currency recognition, the dataset types and the method used are different from this research work. No research is done on the newly released ETB. In this study, the obtained an average training accuracy of 99.7% and average validation accuracy of 99.5% by using an 80/10/10 dataset ratio. Likewise, the model obtained an average training accuracy of 97.8% and average validation accuracy of 99.0% by using a 70/15/15 dataset ratio.

Having high performance on the loss value of the model or the training accuracy value of the model doesn't guarantee high performance on the validation dataset. Since we are running the model on a Central Processing Unit (CPU) the training process takes long hours. The loss value didn't always decrease or the training accuracy didn't always increase it oscillates on each batch size of every epoch.

The performance of the model is highly determined by several factors such as dataset size, quality of the dataset, the ability to tune the high parameters, on the ability of the computer the model can learn fast on GPU processors than CPU processors because of these the model takes hours to learn the data. But the ability to do all the above except the processor ability increases as the experience of the developer increases.

This study has achieved the general objective over the development of the model using CNN,

Transfer learning, Tensorflow's Keras API framework with the preferred outcomes.

6.2. Recommendation

Designing and implementing a currency recognition system is not a simple task. Because of the vastness of the work and the limited resources and expertise, there are still tasks that can be done i n the future to improve or upgrade the system.The following are some of the recommendations that the researcher proposes for future work:

- Increase the number of the data set with good quality.
- Use other types of MobileNet such as MobileNet V2 to obtain better accuracy.
- Perform the training session by changing different parameters and switching the network architecture, different activation functions, loss functions, and an optimization algorithm.
- This model is run on CPU, to get better processing speed run on GPU.
- Port the method to smartphones to develop a robust application for banknote recognition that may help visually impaired people.
- The Keras framework where we have made the model is so suitable for developing mobile applications and embedded systems. So for other researchers, we highly recommend using these models and or datasets and develop a mobile application for these problems.

REFERENCES

[1] Stu Elmes, 'How does machine learning affect your everyday life?', Phrasee. 12-May-2016 [Online]. Available: https://phrasee.co/what-you-need-to-know-about-machine-learning-part-4 how-does-machine-learning-affect-your-life/. [Accessed: 17-Jul-2017]

[2] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuald E. Alsaadi " *A survey of deep neural network architectures and their applications*" Elsevier B.V. on Neurocomputing, pp.234, 2017.

[3] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew "*Deep learning for visual understanding*" Elsevier B.V. on Neurocomputing, pp.187, 2016.

[4] T.D. Pham, Y.H. Park, S.Y. Kwon, K.R. Park, D.S. Jeong, and S. Yoon, "Efficient Banknote Recognition Based on Selection of Discriminative Regions with OneDimensional Visible-Light Line Sensor," In Sensors, Vol. 16, No. 3, pp. 328, 2016.

[5] "National Bank of Ethiopia", http://www.nbe.gov.et/aboutus/legalnotes.html, last accessed on January. 10, 2021.

[6] World Health Organization, "GLOBAL DATA ON VISUAL IMPAIRMENTS 2010," Geneva, Switzerland, 2010.

[7] Yemane Berhane and et al., "Prevalence and causes of blindness and Low Vision in Ethiopia," Ethiopian Journal of Health Development, vol. 21, no. 3, pp. 204-210, April 2007.

[8] Olanrewaju, RashidahFunke, and FajingbesiFawwazEniola. "Automated Bank Note Identification System for Visually Impaired Subjects in Malaysia." 2016 International Conference on Computer and Communication Engineering (ICCCE), IEEE, Kuala Lumpur, Malaysia, July 26-27, 2016, pp. 115-120.

[9] Mittal, S., & Mittal, S. (2018). Indian Banknote Recognition using Convolutional Neural Network. 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU),2018.

[10] Qian Zhang and Wei Qi Yan, " Currency Detection and Recognition Based on Deep Learning" IEEE, 2018.

[11] Takeda, Fumiaki, LalitaSakoobunthu, and Hironobu Satou. "Thai banknote recognition using neural network and continues learning by DSP unit." Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin/Heidelberg, 2003, vol 2773, pp. 1169-1177.

[12] Ch. Ratna Jyothi, Dr.Y.Sundara Krishna, and Dr.V.SrinivasaRao, "Paper currency recognition for color images based on Artificial Neural Network" IEEE, 2016.

[13] Zeggeye, J., and Assabie, Y. (2016). Automatic Recognition and Counterfeit Detection of Ethiopian Paper Currency. International Journal of Image, Graphics and SignalProcessing, 2016.02.04.

[14] Tessfaw, M., Ramani, M., and Bahiru, M. (2018). Ethiopian Banknote Recognition and FakeDetection Using Support Vector Machine. 2018 Second International Conference on inventive Communication and Computational Technologies (ICICCT).

[15] Asfaw Shefraw Alene, "Ethiopian Paper Currency Recognition System" Vol 8, pp 130 IEEE, 2019.

[16] F. García-Lamont, J. Cervantes, and A. López, "Recognition of Mexican banknotes via their color and texture features," Expert Syst. Appl., vol. 39, no. 10, pp. 9651–9660, 2012.

[17] F. Garciá-Lamont, J. Cervantes, A. López, and L. Rodríguez, "Classification of Mexican paper currency denomination by extracting their discriminative colors," Lect. Notes Comput. Sci. (Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8266 LNAI, no. PART 2, pp. 403–412, 2013.

[18] Muhammad Sarfraz, "An Intelligent Paper Currency Recognition" International Conference on Communication, Management and Information Technology 1877-0509 @ 2015 The Authors, Published by Elsevier B. V.

[19] Woo Lee, J.; Hong, H.; Wan Kim, K.; Ryoung Park, K. A Survey on Banknote Recognition Methods by Various Sensors. Sensors 2017,17, 313.

[20] Miseon Han and Jeongtae Kim, "Joint Banknote Recognition and Counterfeit Detection Using Explainable Artificial Intelligence", Sensors 2019.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6719953/

[21] Graham Oppy and David Dowe, 'The Turing Test', in The Stanford Encyclopedia of Philosophy, Spring 2016., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University,2016[Online].Available: https://plato.stanford.edu/archives/spr2016/entriesuring-test/

[22] "Machine Learning", Coursera. [Online]. Available: https://www.coursera.org/learn/machine learning/. [Accessed: 18-Jul-2017].

[23] T. Cover and P. Hart, 'Nearest neighbor pattern classification, IEEE Trans. Inf. Theory, vol. 13, no. 1, pp. 21–27, Jan. 1967. DOI: 10.1109/TIT.1967.1053964

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 'ImageNet Classification with Deep Convolutional Neural Networks, in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp.1097–1105[Online].

Available:http://papers.nips.cc/paper/4824-imagenet-classificationwith-deep-convolutional-neural-networks.

[25] Stacey Higginbotham, 'Facebook is using machine learning to build better computer vision', Fortune. [Online]. Available: http://fortune.com/2015/06/15/facebook-ai-moments/. [Accessed: 29-Aug-2017].

[26] Sambit Mahapatra, ―Why Deep Learning over Traditional Machine Learning?,‖ Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/whydeep-learning-is-needed-over-traditional-machine-learning-1b6a99177063. [Accessed: 15-Feb-2019].

[27] Richard Szeliski, Computer Vision. London: Springer London, 2011, Texts in Computer Science,ISBN:978-1-84882-934-3[Online].Available: http://link.springer.com/10.1007/978-184882-935-0. [Accessed: 18-Jul-2017]

[26] Miseon Han and Jeongtae Kim, "Joint banknote recognition and counterfeit Detection using Explainable Artificial Intelligence",Sensor 2019, 19, 3607; o=doi:10.3390/s19163607

[27] Hubel DH, Wiesel TN (1968) Receptive fields and functional architecture of monkey striate cortex. J Physiol 195:215–243

[28] Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36:193–202

[29] Aerts HJ, Velazquez ER, Leijenaar RT, et al (2014) Decoding tumor phenotype by noninvasive imaging using a quantitative radiomics approach. Nat Commun 5:4006

[30] Lambin P, Rios-Velazquez E, Leijenaar R, et al (2012) Radiomics: extracting more information from medical images using advanced feature analysis. Eur J Cancer 48:441–446

[30] Lee DH (2013) Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks. In: Proceedings of the ICML 2013 Workshop: Challenges in Representation Learning. Available online at http://deeplearning. net/wp-content/uploads/2013/03/pseudo_label_final.pdf. Accessed 23 Jan 2018

[31] Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training GANs. arXiv. Available online at https://arxiv.org/pdf/1606.03498.pdf. Accessed 23 Jan 2018

[32] Liang M, Tang W, Xu DM et al (2016) Low-dose CT screening for lung cancer: computer-aided detection of missed lung cancers. Radiology 281:279–288

[33] Setio AA, Ciompi F, Litjens G, et al (2016) Pulmonary nodule detection in CT images: false positive reduction using multi-view convolutional networks. IEEE Trans Med Imaging 35:1160–1169

[34] Armato SG 3rd, McLennan G, Bidaut L, et al (2011) The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): a completed reference database of lung nodules on CT scans. Med Phys 38:915–931

[35] van Ginneken B, Armato SG 3rd, de Hoop B et al (2010) Comparing and combining algorithms for computer-aided detection of pulmonary nodules in computed tomography scans: the ANODE09 study. Med Image Anal 14:707–722

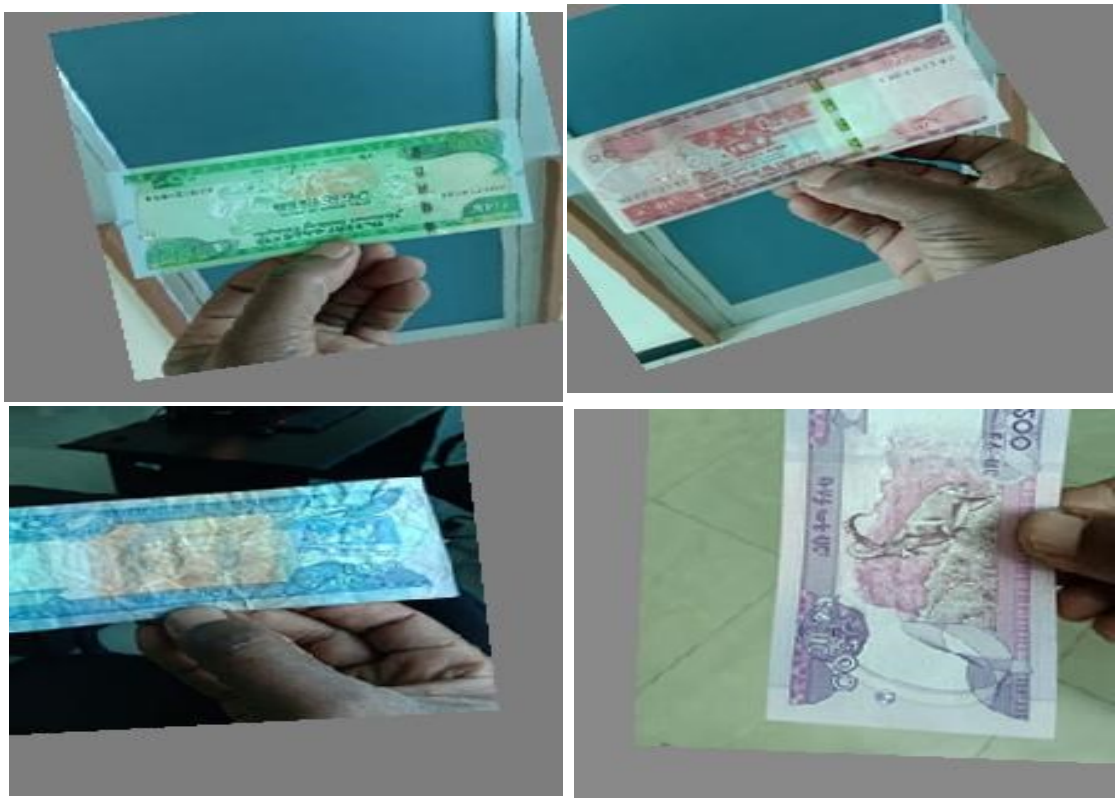
[36] Pedersen JH, Ashraf H, Dirksen A, et al (2009) The Danish randomized lung cancer CT screening trial—overall design and results of the prevalence round. J Thorac Oncol 4:608–614

[37] Kang G, Liu K, Hou B, Zhang N (2017) 3D multi-view convolutional neural networks for lung nodule classification. PLoS One 12:e0188290. https://doi.org/10.1371/journal.pone. 0188290

Appendices

Appendix A: Photo image of Ethiopian newly released Birr

Appendix B: Sample of Augmented Dataset



Appendix C: Model Implementation

```
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.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
```

List of Important Python Library used

Data Augmentation Code

```
datagen = ImageDataGenerator(
        rotation_range=45,     #Random rotation between 0 and 45
        width_shift_range=0.2,   #% shift
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='constant', cval=125)    #Also try nearest, constant, reflect, wrap
#Multiple images.
#Manually read each image and create an array to be supplied to datagen via flow method
```

```
i = 0
for batch in datagen.flow_from_directory(directory='C:/Users/yetimnew/Desktop/tades/ETB_clasification/3/',
                                         batch_size=32,
                                         target_size=(224, 224),
                                         color_mode="rgb",
                                         save_to_dir='C:/Users/yetimnew/Desktop/tades/ETB_clasification_aug/3/',
                                         save_prefix='aug',
                                         save_format='jpg'):
    i += 1
    if i > 55:
        break
```

Setup MobileNet

```
mobile = tf.keras.applications.mobilenet.MobileNet()
mobile.summary()
```

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

```
output = Dense(units=4, activation='softmax')(x)
```

```
model = Model(inputs=mobile.input, outputs=output)
```

```
for layer in model.layers[:-23]:
    layer.trainable = False
```

```
model.summary()
```

Implementation Code

```
model.compile(optimizer=Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
history = model.fit(x=train_batches,
            steps_per_epoch=len(train_batches),
            validation_data=valid_batches,
            validation_steps=len(valid_batches),
            epochs=5,
            verbose=2
)
```

Appendix D: - Model Summary

```
model.summary()
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 225, 225, 3)       0
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (ReLU)            (None, 112, 112, 32)      0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, 112, 112, 64)      256
_________________________________________________________________
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 (BatchNormaliza (None, 56, 56, 64)        256
_________________________________________________________________
conv_dw_2_relu (ReLU)        (None, 56, 56, 64)        0
_________________________________________________________________
conv_pw_2 (Conv2D)           (None, 56, 56, 128)       8192
_________________________________________________________________
conv_pw_2_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_pw_2_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)       1152
_________________________________________________________________
conv_dw_3_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_dw_3_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_pw_3 (Conv2D)           (None, 56, 56, 128)       16384
_________________________________________________________________
conv_pw_3_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
```

conv_pw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 57, 57, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, 28, 28, 128)	1152
conv_dw_4_bn (BatchNormaliza	(None, 28, 28, 128)	512
conv_dw_4_relu (ReLU)	(None, 28, 28, 128)	0
conv_pw_4 (Conv2D)	(None, 28, 28, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_4_relu (ReLU)	(None, 28, 28, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_dw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pw_5 (Conv2D)	(None, 28, 28, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 29, 29, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, 14, 14, 256)	2304
conv_dw_6_bn (BatchNormaliza	(None, 14, 14, 256)	1024
conv_dw_6_relu (ReLU)	(None, 14, 14, 256)	0
conv_pw_6 (Conv2D)	(None, 14, 14, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_6_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_7 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, 14, 14, 512)	4608

conv_dw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_8 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_9 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_dw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_10 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_dw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_11 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 7, 7, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 7, 7, 512)	2048
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0

```
conv_pw_12 (Conv2D)          (None, 7, 7, 1024)        524288
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024)        9216
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 4100
=================================================================
Total params: 3,232,964
Trainable params: 1,867,780
Non-trainable params: 1,365,184
_________________________________________________________________
```