YAREGAL TADESSE TESSEMA

ETHIOPIAN BANKNOTE CLASSIFICATION WITH NEURAL NETWORKS

NEU 2019

ETHIOPIAN BANKNOTE CLASSIFICATION WITH NEURAL NETWORKS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
YAREGAL TADESSE TESSEMA

In Partial Fulfillment of the Requirements for the Degree of Master of Sciences in Software Engineering

NICOSIA, 2019

ETHIOPIAN BANKNOTE CLASSIFICATION WITH NEURAL NETWORKS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By

YAREGAL TADESSE TESSEMA

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering

NICOSIA, 2019

Yaregal Tadesse Tessema: ETHIOPIAN BANKNOTE CLASSIFICATION WITH NEURAL NETWORKS

Approval of Director of Graduate School of Applied Sciences

Prof.Dr.Nadire CAVUS

We certify this thesis is satisfactory for the ward of the degree of Masters of Science in Software Engineering

Examining Committee in Charge:

Prof. Dr. Rahib Abiyev	Departments of Computer Engineering, NEU
Assoc.Prof. Dr.Kamil Dimilier	Department of Electrical and Electronic Engineering, NEU
Assist. Prof. Dr.Boran Sekeroglu	Supervisor, Department of Computer Information System, NEU

To my parents…

ACKNOWLEDGEMENTS

First, I want to thank my Geta for being with me in my every step of life and for everything I got in my life. Next, I want to express my special thanks of gratitude for my supervisor Assist. Professor. Dr. Boran SEKEROGLU who have helped, guide, motivation and mentoring me throughout the hole thesis without him I can't accomplish these. additionally, I want to thank my families for their effortless support and their dedication to love me and support me there pray was always with me in my life and in the accomplishment of these achievement. Finally, I want to thank all my friends all over the world for their support in all those sad times.

ABSTRACT

Due to an increase in financial institutions in Ethiopia the number of banks in Ethiopia have increased and due to this the number of automatic transaction facilities such as automatic teller machines (ATM) and multi-functional counting machines have increased dramatically. For these devices to work properly a banknote classification is a mandatory. In addition to these now a days in Ethiopia there are 1.28 million blind peoples and 2.96 million peoples with low vision now a days these peoples are actively participating in the economic developments of the country but these peoples with vision disability have a lot of problem which prevents them from participating actively in the economic aspects of the country.

In this thesis I studied the Ethiopian banknote and developed three brand new dataset such as the dataset that contains the frontside of the banknotes, the backside of the banknotes and the combination of both banknotes then the dataset is spited in two ways which are one using the 80% training,10% test and 10% development sets and the other one with the proportion of 70% training ,15% test and 15% development then comparison of the two category is performed. Then I developed three ResNet models which are trained on the training dataset which contains 60% and 70% for the two categories, then I tuned the hypermeters to find the best performing model using the development dataset which contains 15% and 10% of the total dataset and finally I tested the model's performance using the test dataset which also contains 15% and 10% for the two categories of the total dataset. I obtained a satisfactory result which is a result for the first model with 70/15/15 proportion splitting of the dataset I obtained a training accuracy of 99.73% and test accuracy of 96.202%, for the second model I obtained a training accuracy of 99.08% and test accuracy of 98.38%, and finally for the third model I obtained a training accuracy 98.82% and a test set accuracy of 96.0869%. for the 80/10/10 proportion splitting of the dataset I obtained for the front model training accuracy of 99.66% and test accuracy of 97.419%, for the backside model I obtained a training accuracy of 99.68% and test accuracy of 99.32%, and finally for the all side model I obtained a training accuracy 100% and a test set accuracy of 100%.

Keywords: Convolution block; Convolutional Neural Network; Identity block; Layer; pooling; ResNet; SoftMax

ÖZET

Etiyopya'daki finansal kuruluşlardaki artış nedeniyle, Etiyopya'daki bankaların sayısı artmış ve bu sayede otomatik para çekme makineleri (ATM) ve çok fonksiyonlu sayma makineleri gibi otomatik işlem olanakları artmıştır. Bu cihazların düzgün çalışması için bir banknot sınıflandırması zorunludur. Bunlara ek olarak bugün Etiyopya'da bir gün var 1.28 milyon kör halk ve şimdilerde az vizyona sahip 2.96 milyon insan bugünlerde bir gün bu halklar ülkenin ekonomik gelişmelerine aktif olarak katılıyor, ancak görme özürlü olan bu halklar ülkenin öneme sahip olduğu bir çok sorunu var onları ülkenin ekonomik yönüne aktif olarak katılmaktan.

Bu tezde Etiyopya banknotunu çalıştım ve banknotların ön yüzünü, banknotların arka yüzünü ve her iki banknotun birleşimini içeren veri kümesi gibi üç yeni veri seti geliştirdim. %80 antrenman, %10 test ve %10 gelişim setleri ve diğer %70 eğitim, %15 test ve %15 gelişim oranına sahip olan iki kategorinin karşılaştırılması gerçekleştirildi. Sonra, iki kategori için %60 ve %70 içeren eğitim veri setinde eğitilmiş üç ResNet modeli geliştirdim, daha sonra toplamın %15 ve %10'unu içeren geliştirme veri setini kullanarak en iyi performans gösteren modeli bulmak için hipermetreleri ayarladım veri kümesi ve son olarak, toplam veri setinin iki kategorisi için %15 ve %10 içeren test veri setini kullanarak modelin performansını test ettim. Veri setinin 70/15/15 oranında bölünmesiyle ilk model için bir sonuç olan tatmin edici bir sonuç elde ettim. I, %99,73 eğitim doğruluğu ve %96,202 test kesinliği elde etti, ikinci model için 99,08 % ve% 98,38 test doğruluğu ve son olarak üçüncü model için %98,82 eğitim kesinliği ve %96,0869 test set doğruluğu elde ettim. %99,66 ön model eğitim doğruluğu ve %97.419 test doğruluğu için elde ettiğim veri setinin 80/10/10 oranlı bölünmesi için, arkadaki model için %99,68 eğitim kesinliği ve %99.32 test kesinliği elde edilmiştir. Son olarak tüm taraflar için %100 eğitim doğruluğu ve %100 test ayar doğruluğu elde ettim.

Anahtar Kelimeler: Evrişim bloğu; Dönüşümlü Sinir Ağı; Kimlik bloğu; Katman; havuzlama; ResNet; SoftMax

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATION

AI:	Artificial Intelligence
API:	Application Programing Interface
ATM:	Automatic Teller Machine
AUD:	Australian Dollar
CB:	Convolutional Block
CNN:	Convolutional Neural Network
Conv:	Convolution
ECB:	Ethiopian Commercial Bank
ETB:	Ethiopian Birr
EU:	European Euro
FC:	Fully Connected
GGD:	Generalized Gaussian Density
HDF:	Hierarchical Data Format
HMM:	Hidden Markov Model
NBC:	Naïve Bayes Classifier
NN:	Neural Network
OPENCV:	Open Source Computer Vision
ORB:	Oriented FAST and Rotated BRIEF
PCA:	Principal Component Analysis
QWT:	Quaternion Wavelet Transforms
ResNet:	Residual Neural Network
ReLU:	Rectified Linear Unit
SGD:	Stochastic Gradient Descent
SAR:	Saudi Arabian Riyal
SURF:	Speed-Up Robust Features
SVM:	Support Vector Machine
USD:	United States Dollar

CHAPTER 1

INTRODUCTION

1.1. Background

Artificial Intelligence (AI) is the technology currently transforming every major industry ranging from manufacturing, communication, healthcare, transportation and many more to day we see a surprisingly clear path for AI to bring about a big transformation as to the society and off curse with the part of AI that is rising rapidly and one of the most those rapidly rising parts of AI is deep learning. Deep learning is already transformed all of the traditional internet business available like advertising and web search but deep learning is also being used currently in many new areas in ways of helping peoples to be created everything ranging from butter health care to delivering personalized education and to precision agriculture to even self-driving cars and banknote recognition, counterfeit identification, genuine banknote identification and banknote classification are also one of the fields where machine learning is getting a great success.

Money classification system are playing a crucial rule in almost all of automated transaction facilities such as Automated Teller machine (ATM) and multifunctional counting machines which are making our day to day activity easier these all machines must need the software that are capable of efficiently identifies and classifies a given banknotes. There are a number of proposed methods for banknote counterfeit identification, recognition and classification (Lee J et al, 2017) Some of them are Principal Component Analysis (PCA), Hidden Markov Model (HMM), Naïve Bayes Classifier (NBC) and Neural Network (NN), a number of researchers have used these different techniques to identification, recognition and classification different country banknotes for instance the method proposed by (Gai et al, 2013). Uses quaternion wavelet transform (QWT) and Generalized Gaussian Density (GGD) for feature extractions and neural network for classification of banknote images of USA Dollar (USD), China yuan (CHY), European Euro. And Pham et al proposed a method to classify banknotes from USD, south African Rand (ZAR), Angola Kwanza (AOA) and Malawian Kwacha (MWK) using K-base mean classifier (Pham et al, 2016). (Bhurke et al, 2015) proposed a Euclidian distance-based banknote recognition and classifier on the bank notes of Indian Rupee (INR), Australian Dollar

(AUD), EUR, Saudi Arabian Riyal (SAR) and USD this will be discussed in chapter two in more detail.

The banknote recognition and classification systems (Lee et al, 2017). developed earlier for different countries are done on different image datasets but there are rear researches that have been done on the Ethiopian banknote named Birr classification because of these there is no image dataset or a model that uses machine learning to classify the Ethiopian banknotes. so, the main aim of my paper was to study the Ethiopian banknotes in detail and prepare a machine learning model that can be able to classify the Ethiopian Banknotes. In the study I select the appropriate neural network architecture for the model, study and identify the parameters and hypermeters that fit the model and make the model perform efficiently and finally develop a dataset, train the model using the dataset and test the performance of the model.

In today's technology deep machine learning is capable of doing a lot of thing like route finding, robotics, patient expert systems. In developing countries like Ethiopia machine learning is known by just its name there is no any practical applications made that can significantly change the society life. But still a number of problems exist that can be solve with the help of deep machine learning specially in the economic sector developing countries like Ethiopia are highly vulnerable to fake money and since we have around 100 million population the number of peoples a bank service is increasing so now it's time to use machine learning to solve those problems. Additionally, in a country where there are millions of peoples are blind the role of machine learning plays crucial role to keep those group of peoples to participate in social, economic and political aspects of the country.

1.2. Statement of the Problem

currently in Ethiopia there are 1.28 million blind peoples and 2.96 million peoples with low vision (Berhane et al 2018). which means there is a number of propels who loses their eye sight for a number of reasons those group of peoples are starting to participate actively in the current social, economic and political aspects of the country because an advancement in a lot of technologies and the encouragement for those disability groups by the government to those group of peoples. The big problem for those group of individuals to participate in economic aspect of the country is there is no any machine that can help them identify bank notes.

identify banknotes because of their blindness disability which makes their life hard especially from the economic perspective the study we made and the final model I develop can may be embedded in different automatic transaction devices and can help those disabled group of peoples.

Additionally, currently in Ethiopia there are a number of automated transaction facilities like automated teller machine (ATM) and multifunctional counting machine. All over the country in banks in the county. Specially the automated teller machine is used by every citizen in the country (ET Switch,2019) its ease of use and availability in many places and at any time for the customers use which makes life easier to a number of citizens by allowing to withdraw money at any time they want in almost near to them, transfer money to the friend, family or business partner easily within sort time, But on the other side 80% of the country population is having his income by agriculture and small local trades with the low income and these populations are illiterates which do not have that much education but still most of them are using banks because banks are highly spread over the country that makes them to be familiarly with banks especially the Ethiopian Commercial Bank (ECB). But the problem here is that these farmers got the money not in an arranged order of the Banknotes in their increasing order like (all the 100 liras together next all the 50 liras together then…) because in the market everybody don't have time to do so and don't have enough knowledge to arrange them systematically during collecting the money and in the worst case they may don't know the values of each notes and again the big problem is that all of the bank's ATM machines of all the banks the country are only used to withdraw money they are not used to deposit money as the ATMs here. So, in Ethiopia it's the accountants of the bank duty to arrange the money in order when the farmers bring them for deposit. So, what I am proposing is that using machine learning (neural network) we can develop a Machin learning (neural network) model that can enable as to classify all of the Ethiopian Banknotes so that it may be deployed and used by different automated transaction facilities such as ATM machines or multi-functional counting devices so that anyone can deposit money on the ATM machine or the bank accountant can easily classify the Banknotes in to their category.

1.3. Research Motives

in Ethiopia the automatic transaction facility like ATMs are used only to withdraw money from an account using the ATM card and get information about your account and also transfer money from one account to the other account so I thought that is the only use of ATMs when I was back in home Ethiopia but after I arrive here in Cyprus ATM machines are used to deposit money to an account as well and then in my first semester in Near East University I take the course called soft computing and in this course there is a chapter named Neural Network that teaches me that it's possible to train machines using machine Learning(Neural Network). From that day I was thinking to design a neural network model, prepare the datasets for the Ethiopian banknotes train the model and classify the different Ethiopian Banknotes (which are note 5,10,50 and 100) using the model so that it can be integrated on different automated transaction facilities like ATM machines and help the society and the banks.

1.4. Objective of the Study

1.4.1. General objective

The general objective of this study is to study the Ethiopian banknotes develop a model that enables us to classifies the banknotes.

1.4.2. Specific objective

To achieve the general objective, I mentioned the research needs to address these specific objectives mentioned bellow

- To select the appropriate neural network architecture
- Develop the model
- Develop the datasets (training, development and test datasets)
- Evaluate the performance of the model using the appropriate measurement metrics.

1.5. Research Questions

The questions that this research is going to answer are as following

- Which Neural network architecture is appropriate for the Ethiopian Banknote classification model.?

- How to create datasets that appropriately tests the model created.?
- What are the values of the hypermeters that are used in the model.?
- How to train the model we developed with the dataset created.?
- What is the efficiency of the model when tested.?

After the end of the study all of the above questions will be addressed. So, these studies are generally designed to answer all of the questions I mentioned above on completion.

1.6. Scope of the Study

The scope of these study is to study the Ethiopian Banknote which is called Birr that contains currently 4 notes which are five(5), ten(10),fifty(50) and one hundred(100) and select the appropriate neural network architecture then prepare the model, develop datasets(training dataset, development dataset and test dataset), train the model using the training dataset, find the appropriate parameters and hypermeters of the model by tuning different values using the development dataset and finally test the performance of the final model using the appropriate metrics on the test dataset.so these paper don't include fake banknote detections.

1.7. Significance of the Study

In current technology days a number of transactions are performed using computers in Ethiopia and science 80% of the Ethiopian population life by agriculture and almost all of them are illiterate or with no or little education which can got money and uses the bank accountants to arrange and count their money for them and all bank ATM machines and automated transaction facilities do not allow to deposit money so developing a model that can recognize banknotes and classify them so they can be integrated with ATM machines or any automated transaction facility and enable everybody to use the ATM machines or any automated transaction facilities to deposit money, to classify the notes which can may help to ease bank processes.

In addition to these in today's technology world the number of peoples participating in the market or in any social ,political and economic aspects of the community the participation of those peoples which have eye blindness disability are increasing from day today and as human they are capable of making money and using money to get services but they don't have the capability to recognize and classify the banknotes so the coming of these banknote recognition

and classification model may enable those individuals with blindness disabilities to participate and play a crucial in the economic aspect of the country.

1.8. Expected Result

At the end of these study all the research questions that I have mentions in the research questions part of these pare are all addressed and a working Ethiopian banknotes recognition and classification Neural network model will be developed.

1.9. Organization of the Document

This research paper is organized into six (6) chapters including the current chapter one which is introduction. Chapter two deals about the literature review about the machine learning in depth and related works that have been made about the Ethiopian banknotes and other bank notes from different countries. In chapter three we will discuss in detail about the concepts that we have used in the convolutional neural network. In chapter four we discuss about the methodology we have used to accomplish the research work and in chapter five we will see the implementation of the research which includes the preparations of the dataset and the model in detail including the architecture we used, in chapter six we will see about the results we got on the model and discuss in detail why and how we got those results. And finally, in chapter seven we will see recommendations and conclusions.

CHAPTER 2

LITERATURE REVIEW

2.1. Overview

In this chapter first the researcher will discuss related works done by other researchers on bank note classification and how they are done next the researcher will discuss in detail what is machine learning, why machine learning, what is deep learning then what is Neural network (NN) and what is CNN then I will describe what is banknote classification and how it is done using Convolutional Neural Network finally

2.2. Literature Review

2.2.1. Literature review on Ethiopian banknotes

Now let's see studies that are made on Ethiopian Banknotes (Zeggeye et al, 2016) proposed Automatic Recognition and Counterfeit Detection of Ethiopian Paper Currency in their study they proposed method have three parts one is currency image acquisition which is about getting the target picture the second one is currency denomination which is achieved by processing of the image ,extracting the characteristic features from the image and classifying the image based on those features in the third faze counterfeit detection is performed uses feature extraction of image processing to classify the images in his approach he tried to classify the bank notes based on feature extractions. Another study on the counterfeit currency identification has been made by (Raimond et al, 2017) also ureses feature extraction of image processing and performs counterfeit identification. On other study (Tessfaw et al, 2018) have done Ethiopian Banknote Recognition and Fake Detection in their study they used Support Vector Machine (SVM) to recognize and detect the Ethiopian banknotes from other country banknotes. In another study (Jegnaw Fentahun, 2014) proposed an automatic recognition of Ethiopian paper currency Jegnaw in his study used a currency characteristic comparison which uses feature extraction which is the process of transforming rich contents of the image in to a number of various content features and those features are features with are relatively higher classifying or discriminating ability and ignoring those features with low discriminating or classifying ability. As the knowledge of the researcher these are the researches that have done on Ethiopian bank notes and nearly no research study is made on the classification of Ethiopian banknotes on a complete

base of using Convolutional Neural Network (CNN), I have to study further studies which are made all over the world so I tried to study them and presented them in the next topic.

2.2.2. Literature review on other country banknotes

In practice automated transaction facilities like ATM machines and multi-functional counting devices have capable of processing a number of tasks such as denomination determining, counterfeit classification, fitness classification on various country banknotes (Lee J et al, 2017) In the past years the most popular approach that is used most of the time for banknote recognition and classification is image processing using feature extraction (Bala N et al, 2014). But concerning banknote recognition and classification there is a number of studies have been already made from different countries on completely deferent data (unique) image datasets of each countries banknotes and using a different way. For instance, the method proposed by (Gai et al, 2013). Uses quaternion wavelet transform (QWT) and generalized Gaussian density (GGD) for feature extractions and neural network for classification of banknote images of USA Dollar (USD), China yuan (CHY), European Euro. And Pham et al proposed a method to classify banknotes from USA Dollar (USD), south African Rand (ZR), Angola Kwanza (AOA) and Malawian Kwacha (MWK) using K-base mean classifier (Pham et al, 2016). (Bhurke et al, 2015) proposed a banknote recognition and classifier on the bank notes of Indian Rupee (INR),Australian Dollar(AUD)Euro(EUR), Saudi Arabian Riyal (SAR) USA Dollar(USD) in their proposed method they performed an image preprocessing which transforms the image in to the needed form such as change to greyscale, segmentation are performed next boundary detection is performed and the regions of interest (ROI) is extracted next the desired feature are extracted and the comparison between the extracted feature and the ideal calculated feature values is performed and displayed the output.

Additionally, (Kwon et al, 2016). have done similar study on the recognition and classification of banknotes from USA Dollar (USD), Indian Rupee (INR) and Korean Won (KRW). (Khashman et al, 2005) in their study the recognition and classification of two different banknotes which are the Cyprus pound and the Turkish Lira. in another study (Rashid et al 2015) proposed support vector machine (SVM), artificial neural network and hidden markhove model (HMM) to recognize and classify USA Dollar (USD) and Euro banknotes and (Youn et al, 2015). also proposed a multi – country banknote classification methods. a study by (Dunai et al, 2017) have also proposed detection and recognition of Euro banknotes the proposed method

uses Haar technique (Viola et al, 2001) in order to locate the zones of interest in a given image instead of analyzing all pixels of a given image so that can help to reduce the computation time after the Haar technique AdaBoost algorithm is applied for identifying the euro banknotes and make the classification once the euro banknotes are recognized by the AdaBoost algorithm the Speed-Up Robust Features(SURF) (Viola et al, 2001) algorithm is used to identify interest points in the image with their own characteristics once the interest points are and their neighboring points are was identified the SURF descriptor is fetched from the region and at the end the features are matched between the trained image and the image gotten by the system, another paper by (Safraz M,2015) have proposed an Intelligent Paper Currency Recognition System which is based on interesting features and correlation between images the steps of processes performed in these study are first the banknotes are collected, then the collected banknotes are scanned, then image processing is performed such as noise removal, resizing image ,changing the image to grayscale and etc.. then the feature extraction phase continues and finally weighted Euclidian distance and radial basis neural network is applied for the classification of the image.

In other study by Prof Adnan Khashman and Asist.prof.Dr Boran have used The combination of both image processing and neural networks for recognition of various banknotes specially Turkish Lira and Cyprus Pound

(Mohammad et al ,2014) proposed Recognizing Bangladeshi Currency for Visually Impaired. Mohamed in his paper used Oriented FAST and Rotated BRIEF (ORB), a method which is developed in the lab of Open Source Computer Vision (OPENCV) because ORB is faster than the SURF and SIFT. In another paper Tuyen (Danh Pham et al,2018) proposed a multination banknote classification system which is based on the method of visible-light banknote image captured by a one-dimensional line sensor they also have used convolutional neural Network for the classification purpose and they performed the experiment on a combined banknote image databases that includes six country banknotes which gives 64 denominations, Another study by (Mittal et al, 2018) have proposed banknote recognition based on selection of discriminative regions with one dimensional visible light sensor their proposed method contains acquisition of image and preprocessing such as segmentation and normalization next the regions with high power of discrimination are selected using a similarity map next with the data selected with similarity map using principal component analysis (PCA) then the optimal reduced feature

vector extracted and finally the banknote dienamines and the direction of the input image is determined by applying the K-mean algorithm on the previous PCA features, another study by (Tuyen et al, 2017) have proposed multinational banknote classification based on visible light sensor and convolutional neural network(CNN) the proposed banknote classification is performed on six(6) countries banknotes such as Chinese Yen (CNY), Euro, Japanese Yen(JPY), Russian Ruble(RUB), South Korean Won(KRW) and USD on the methods they used two different Neural Networks one for detecting on side of the note and the other for the other side of the note then score level fusion is performed which takes the average of the two neural networks average and finally SoftMax classification is performed on the averaged output to make the classification.

but not only the above studies there are also a number of studies made on different countries of the world. Here want I noticed is that most of the banknote recognition classifications studies that are made are using feature filtering of image processing and some of them uses neural networks to classify their country banknotes.

CHAPTER 3

CONVOLUTIONAL NEURAL NETWORKS

3.1. Neural Networks

Neural network is a computer system modeled on the human brain and nerve system. The Neural Network approaches a problem in a way that is to take a huge number of labeled data which contains training set, development set and test sets then develop a system that can learn from the giving labeled training set and automatically infers a rule that enables to solve the problems in hand in addition to this if we give the neural network a larger number of training set then the neural network can learn more complex problems of the given problem with a higher accuracy.

3.1.1. Perceptron

perceptron's are developed in 1950 by Frank Rosenblatt and perceptron is one type of artificial neuron that takes a number of binary (1 or 0) inputs (X_1, X_2, X_3……. Xn) and outputs a single binary (0 or 1) as shown in the Figure 3.1 below.

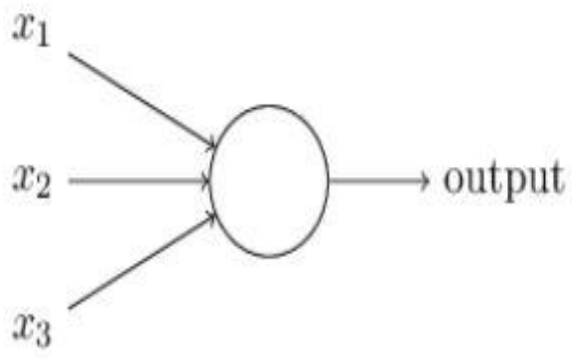

Figure 3.1: Single neuron

As shown in the Figure 3.1 above the neuron takes 3 binary inputs and produces a single binary output. Then Reosenblatt proposed a simple rule that helps to calculate the output given the binary inputs so he introduced a weight w_1, w_2, w_3, …., wn where n is the number of input features to the perceptron and they are real numbers expressing how much important the respective inputs are to the output. So, the perceptron output is 1 or 0 depending on the $\sum_j W_j X_j$

is less than the given threshold or greater than the given threshold which is mathematically given by the following equation 3.1

$$output = \begin{cases} 0, \ if \ \sum_j W_j X_j \leq 0 \\ 1, \ if \ \sum_j W_j X_j > 0 \end{cases} \quad (3.1)$$

So, by combining a number of such perceptron we got these big network

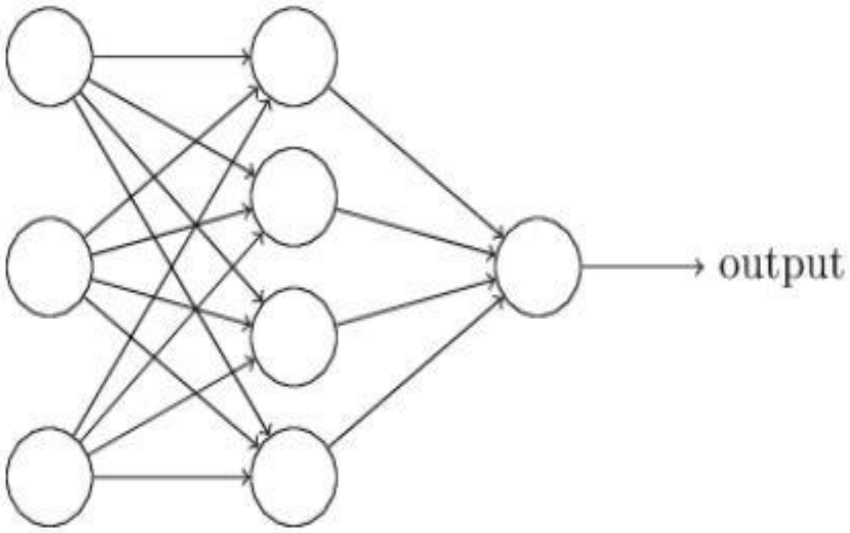

Figure 3.2: Network that combine perceptron's

In this network shown in Figure3.2 above we call the first column of the network the first layer and the second columns the second layer of the network etc.… now we introduce new parameter called Bias(b) it's the measure of how easy is it to the perceptron's to get the output a one (1) value. So, our output equation will be come

$$output = \begin{cases} 0, \ if \ \sum_j W_j X_j + b \leq 0 \\ 1, \ if \ \sum_j W_j X_j + b > 0 \end{cases} \quad (3.2)$$

Now we have a Neural Network (NN) of perceptron neurons that we use to solve our problem in our case that can give image pixel and classify into its respective class so we expect the Neural Network to learn the parameters such as the weight(W) and base(b) that helps the network to correctly classify the banknotes to do so what we want is for every small changes on Weight(W) or base(b) on our Network can only cause a small corresponding change in the

output of the Neural Network so that these will make learning possible but this is not what happen when our Neural Network is made of perceptron neuron what happens is that when we change weight(W) or base(b) of one perceptron neuron on the network in a small value it can may completely flip the output of the perceptron say from 1 to 0 or 0 to1 and that flip may cause the hole Network to behave in a different and very complicated way. To overcome these problems new type of artificial Neuron is introduces which is called the Sigmoid Neuron which is similar to the perceptron neuron but modified in a way that a small change in Wight (W) or base(b) of a neuron in the network can only cause a small change to their output which makes the Sigmoid neuron to learn. Basically, what makes the Sigmoid neuron from the perceptron is that it applies a function named sigmoid function to the output of the perceptron neuron $(W * X + b)$ and now we got the new output which is $\sigma(output)$ and these functions are called activation functions

$$\sigma(z)=\frac{1}{1+e^{-z}} \qquad \text{where } z=(W*X+b) \qquad (3.3)$$

$$\sigma(z)=\frac{1}{1+e^{\left(-\sum_j W_j X_j - b\right)}} \qquad (3.4)$$

here in the equation 3.3 and 3.4 above the sigmoid is similar to the perceptron except that its output is not exactly 0 or 1 instead a number of values between 0 and 1 these will be shown in Figure 3.3 below.

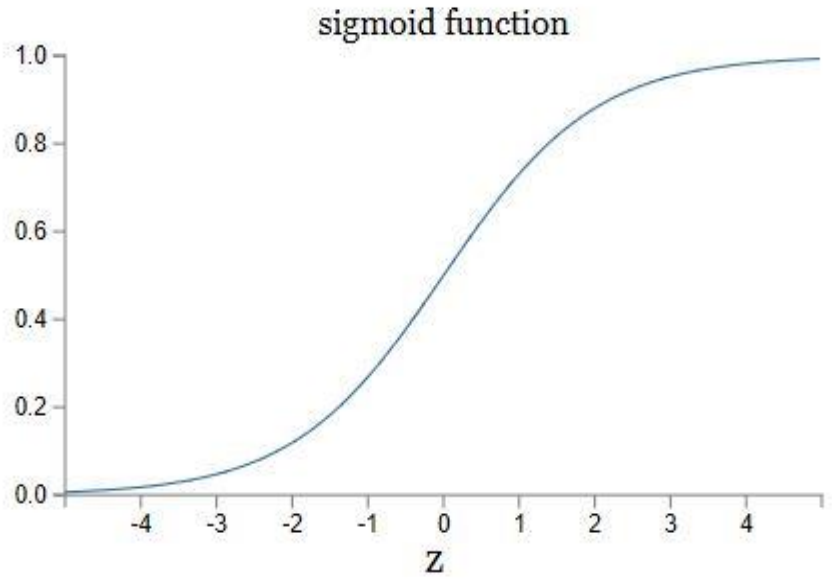

Figure 3.3: Sigmoid function output

As shown in the Figure 3.3 shown above the output value is a value given between 0 and 1 not exactly 0 or 1 it's any value which is between 0 and 1.

There are also other familiarly activation functions one that always works better than sigmoid is the tanh (Hyperbolic tangent function).

$$a = \tanh(Z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \qquad (3.5)$$

This is actually the shifted version of the sigmoid function and it works butter than sigmoid function. So tanh activation function is that the output values are not between 0 and 1 they are values between 1 and minus 1as shown below in Figure 0.4 which means the mean of the activation that comes out the hidden layers are close to having mean zero these makes learning for the next layer a little bit easier. The one exception that we have to use sigmoid is when the output of the neural network is 0 or 1 which is binary classification, we have to use sigmoid activation function only at the output neuron since the output have to be between 0 and 1

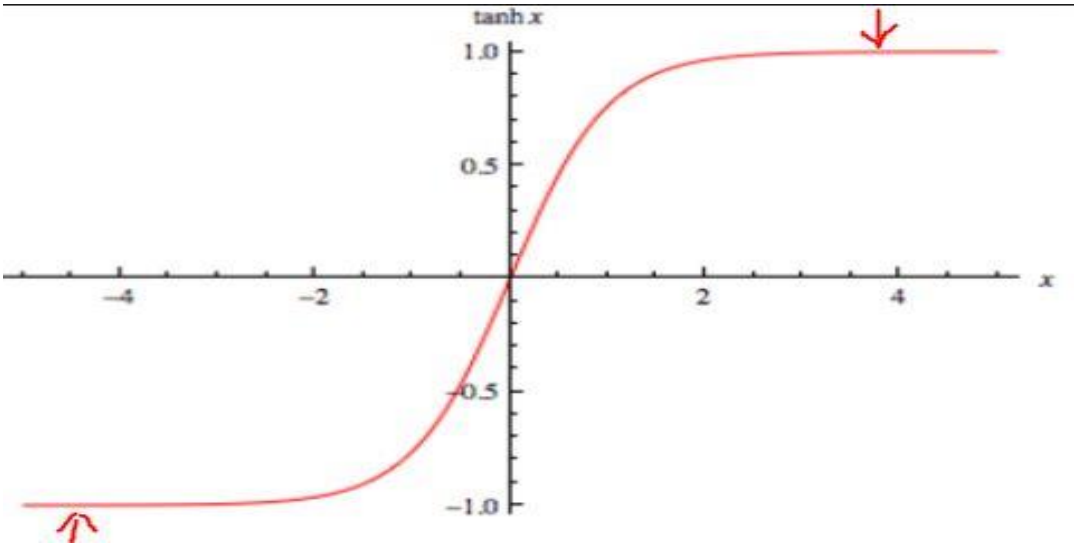

Figure 3.4: Tanh activation function

As shown in the Figure 3.4 above shows that the output value of the network will be between one and negative one which means that the thanh function have a maximum value of one or a minimum value of -1 instead of having a value of zero or one or instead of having a continues value between 0 a one like the sigmoid function which have a value of between negative one and positive one.

As shown in the Figure 3.4 the main disadvantage of both sigmoid and thanh functions are that as the value of z is very large or small the slope or the derivative or the gradient of the functions is become very small which is close to zero as show on Figure3.4 above and these will make the gradient descent or the learning process very slow. So, on the other hand one popular method for activation function is used to solve this problem this is to use the Rectified Linear Unit (ReLU) activation function as shown below in Figure 3.5 and equation 3.6 the function makes the slop or the derivative of the function to be 1 as long as z is positive and 0 when z is negative

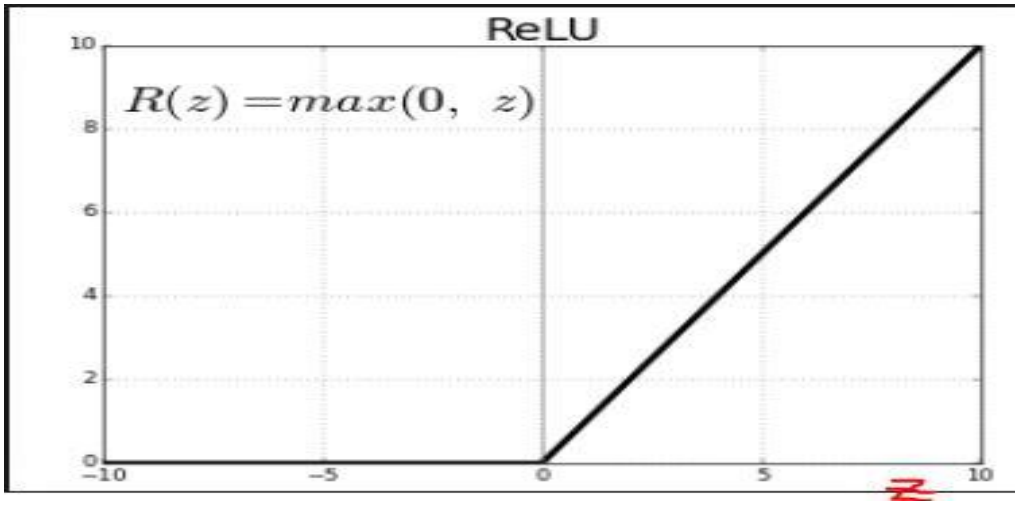

Figure 3.5: ReLU activation function graph

And the function for the ReLU activation function is given as

$$a = Max(0, Z) \qquad (3.6)$$

3.1.2. Architectures of the neural network

Neural Networks have three groups of layers one is the input layer, second is the Hidden Layer the number of layers in the hidden layer depends as the problem but the number of layers I the hidden layer is one or more and the third is the output layer but when we define the number of layers a given network have most of the time, we don't count the input layer, we only count the hidden and output layers as shown in the Figure 3.6 below

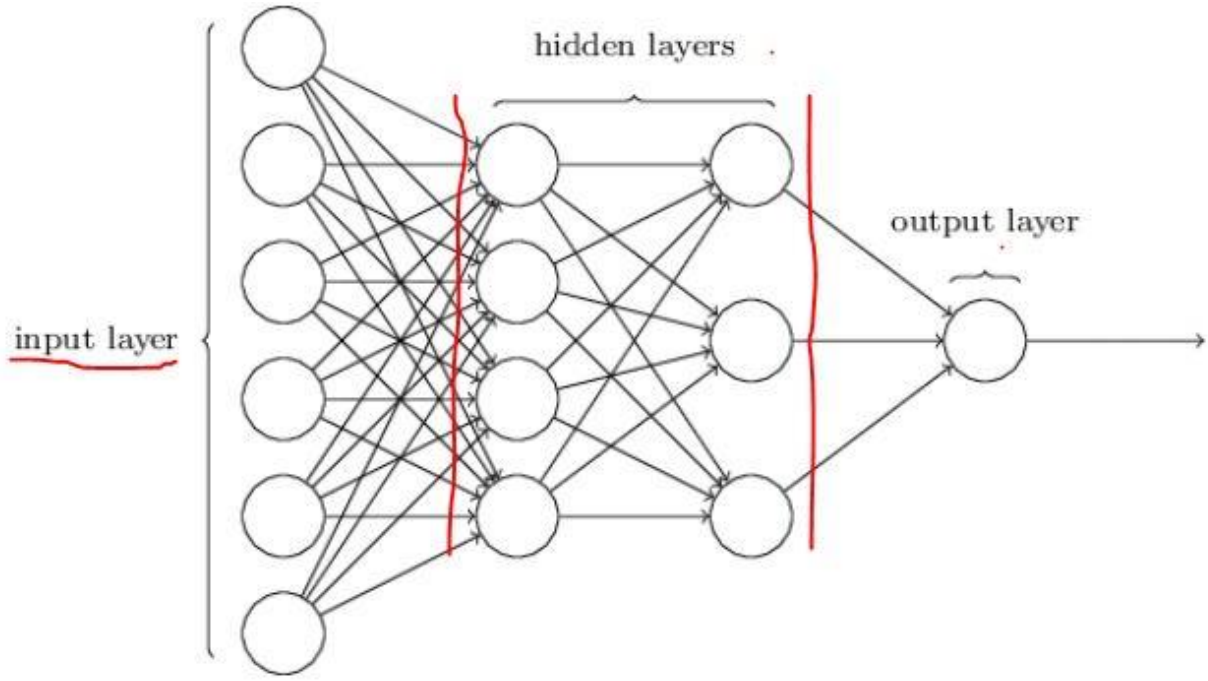

Figure 3.6: Neural network architecture example with 3 layers

So, the number of layers the given Neural Network in Figure 3.6 of the above have 3 layers which are the 2 hidden layers and the 1 output layer the neural network above shows that the output of one layer neurons is feed as input to the next layer neurons such neural network is called Feed Forward neural Network there is another type of neural Network where in which feedback loop is possible the idea of such neural networks is that the neurons are fired only for a limited amount of time and become deactivated during their activated period they stimulate other neurons which will be fire latter for a limited time that causes another neuron to be fired and it continuous so as we go like these we got a cascade of neurons firing so in such model a loop can't be a problem because the output of a neuron only affects its inputs latter not instantaneously

3.1.3. Phases in neural network

3.1.3.1. Parameter initialization

To implement a neural network that solves our problem we have to initialize the parameters that are going to be learned by the neural Network but before we our neural networks starts to learn

those parameters weight(W) and base(b) they must have first to be initialized there are a number of different initialization techniques used.

1. **Zero Initialization**: - these is initializing all the parameters Weight(W) and base (b) to an initial value of zero.
2. **Random Initialization**: - initializing the network parameters weight(W) and base(b) to some random value.
3. **He initialization:** - is the set the variance of the parameters specially weight parameter to $\frac{2}{n}$ where n is the number of input features going into the neuron to make the variance of the parameter use the equation below

$$w^L = np.ranodom.randn(Shape) * \sqrt{\frac{2}{n^{l-1}}} \qquad (3.7)$$

These can also help to solve the gradient vanishing or the gradient exploding problems in deep neural networks

4. **Xavier Initialization**: - the Xavier initialization is similar to the He initialization but it sets the variance of the parameters to $\frac{1}{n}$ instead of $\frac{2}{n}$ and the equation is given below.

$$w^L = np.ranodom.randn(Shape) * \sqrt{\frac{1}{n^{l-1}}} \qquad (3.8)$$

5. There is also rarely used initialization technique proposed by Yosha Bengio and his colleagues which uses the equation below

$$w^L = np.ranodom.randn(Shape) * \sqrt{\frac{2}{n^{L-1}+n^L}} \qquad (3.9)$$

3.1.3.2. Propagation

There are two types of propagation in Neural Network such as forward propagation and backward propagation

a) **Forward Propagation**: - as its name indicates forward propagation is performing every neurons function in every layer of the Network up to the end until we find the Y hat or the output and compute the cost for m number of examples in the training set the operation performed by one neuron is described in the Figure 3.7 Below.

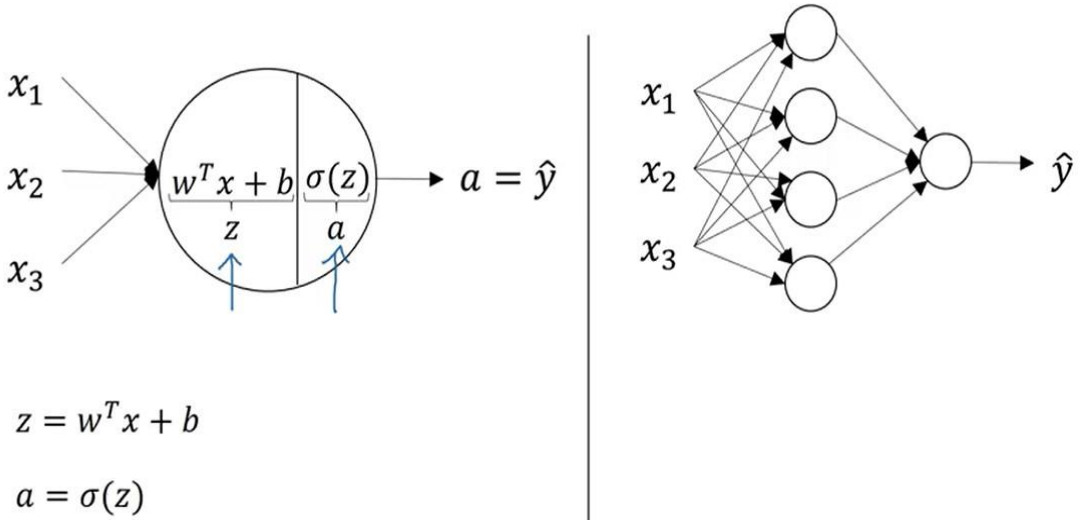

Figure 3.7: Operation performed in a single neuron unit

As shown in Figure 3.7 above the left figure shows operations performed in one neuron there are two operations one is finding $Z=W^TX+b$ and next is performing the activation here we see the sigmoid function as an example but it can be any activation function tanh, ReLU or leaky ReLU. And the right of the figure we see a number of such single neurons are combined in layer to layer to form the big Neural Network and produce an output $\hat{Y}$. So, we perform this operation we see on the single neuron to every neuron in each layer moving forward until we found our $\hat{Y}$ and this is called Froward Propagation.

b) **Backward Propagation**: - in the backward propagation we have to use the $\hat{Y}$ we got in the forward propagation and the Y label in the dataset use the cost formula and find the derivative of the cost function with respect to Z,W and b which is dZ,dW and db respectively for all the layers in the network and make gradient descent for the parameter w that will help as to make the gradient descent or the learning of the parameters the calculations performed in each

layers of the neural network in the forward and backward propagation are summarized in the Table 3.1 below.

Table 3. 1 : examples of backward and forward propagation operation

Forward Propagation	**Backward Propagation**
$z^1 = w^1 X + b^1$	$dZ^L = A^L - Y$
$A^1 = g^1(Z^1)$	$dW^L = \frac{1}{m} dZ^L A^{L-1}$
	$db^L = \frac{1}{m}\sum_{i=1}^{m} dZ^L$
	$W^L = W^L - \alpha * dW^L$
	Maybe little different if we use other optimization algorithms like momentum, RmsProp and Adams algorithm
$z^2 = w^2 A^1 + b^2$	.
$A^2 = g^2(Z^2)$	.
.	.
.	$dZ^2 = A^2 - Y$
.	$dW^2 = \frac{1}{m} dZ^2 A^1$
.	$db^2 = \frac{1}{m}\sum_{i=1}^{m} dZ^2$
.	$W^2 = W^2 - \alpha * dW^2$
	$b^2 = b^2 - \alpha * db^2$
$z^L = w^L A^{L-1} + b^L$	$dZ^1 = W^{[2]T} dZ^{[2]} * g^{[1]'}(Z^{[1]})$
$A^L = g^L(Z^L)$	$dW^1 = \frac{1}{m} dZ^{[1]} X^T$
	$db^1 = \frac{1}{m}\sum_{i=1}^{m} dZ^1$
	$W^1 = W^1 - \alpha * dW^1$
	$b^1 = b^1 - \alpha * db^1$

As we see from the Table 3.1 the after the forward propagation the backward propagation continues and the propagations are performed on the entire training set (mini batch set of the training set) once because of vectorization.

3.2. Convolutional Neural Networks (CNN)

Convolutional Neural Networks are similar to the other Neural Networks with some differences one is that CNN are made with one assumption that their inputs are images the other basic difference is that lets take an image of size 32x32x3 a single fully connected neuron in the first

layer of regular neural Network would have 32x32x3=3072 weights and an image with 200x200x3 will have 120,000 weights and since we have a number of such neurons in each layer it gets high as we add up them and then these quickly leads to overfitting so CNN takes advantages of these because in CNN the layers are arranged in 3D width, height, depth dimension and a neuron on a given layer is connected to only a small region of the layer before it rather than all of the neurons in a fully connected way. All the layers of the CNN transform the 3D input into a 3D output

3.2.1. Layers that make up convolutional neural networks

The convolutional neural network is made of a number of different types of layers such as the conv layer or the convolution layer, thee pooling layer and finally other layer that contains the fully connected layer, flatting layer and SoftMax classification layers as one these all layers have their own different forms and their own different purpose as described below.

3.2.1.1. Conv layer

conv layers (convolution layers) are the core building blocks of the convolutional Networks which performs most of the computational heavy lifting. These layer have also learnable filters which contains parameters those each filter relates to the small position of the input volumes width and height but they extend the full depth of the input volume for example for an input volume of size 120x120x3 image the filter size may be 5x5x3 volume from the above example the depth of the filter and the input volume is the same which is 3 but the filter have the small portion of the width and height of the input volume so as we pass forward we slide/convolve the filter across the width and height of the input volume then on each portion do an element wise dot product between the filter and the small part of the input volume then sum them up and put the sum to a single point of our 2- dimensional output that gives the response of that filter at very special position instinctively the filters that activate when they see some type of visual feature and we have a number of such filters in our conv layer and each will produce a different 2 – dimensional activation map and by stacking those 2-deminsional outputs along the depth dimension we get the output volume which have the same depth as the number of filters we have and its width and Hight is determined by two(2) parameters named Stride and padding.

A. **Stride: -** these defines the stride with which we slide the filters on the input volume example Stride=2 means we slide the filter on the input volume 2 pixel at a time in the width direction and 2 pixels at a time in the height dimension of the input volume.it is not must to use the same stride value to the width and height we can may use different width and height values for example we can may use 3,2 which means slide the filter 3 pixel in the width direction and 2 pixel in the height direction and Increasing the Stride value decreases the output volumes depth and height as we will see in the formula below 3.10. For multi-dimensional input volumes like RGB images (Read Green Blue) which have three volumes one for the read, one for green and one for the blue the filter will also have the same number of volumes as the input volume if the input volume has three sizes the filter will also have three filters so the corresponding filters will be convolved with their corresponding channel in the input volume.

B. **Padding: -** there are two reasons for using padding the first one is it helps to have deep Networks. Because if we don't use padding every time, we apply a convolution operation the input volume shrinks by some value so after some conv layer the input volume gets smaller and smaller which prevents us from designing deep Networks. The second is the pixels at the corner edge of the input volume are touched small times but pixels in the middle of the input volume are used many times so we are ignoring away a lot of information at the corner edge of the input volume to solve these problem we use padding which adds an additional border pixel over all the edges of the input volume for example if we have 6x6x3 input volume and if we pad with padding =2 we end up with 8x8x3 input volume. as seen in Figure 3.8 below.

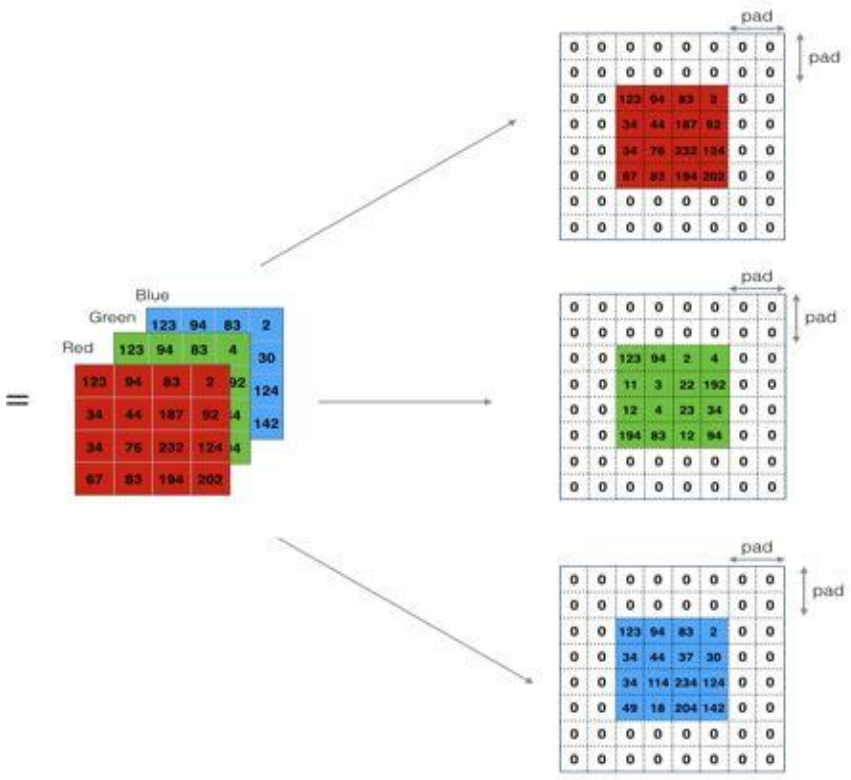

Figure 3.8: Padding example for RGB image with padding=2

So, with these paddings and strides the size of the output volume will be given as the formula below.

$$width = hight = \frac{n+2*padding-f}{Stride} + 1 \qquad (3.10)$$

$$where \quad f - \quad filter\ and\ n - number\ of\ filter$$

3.2.1.2. Pooling layer

The main aim of the pooling layer is to overcome the problem of overfitting by reducing the amount of parameters and computation in the network one of the pooling technique is max-pooling which is a 2x2 or some x by x dimension we put on every depth slice of the input volume and take the maximum over those 4 numbers which can almost reduce the 75% of the activations there is another pooling called average pooling which takes the average of those x by x regions as shown in the Figure 3.9 below and one basic thing in pooling layer is that there is no any parameter to learn.

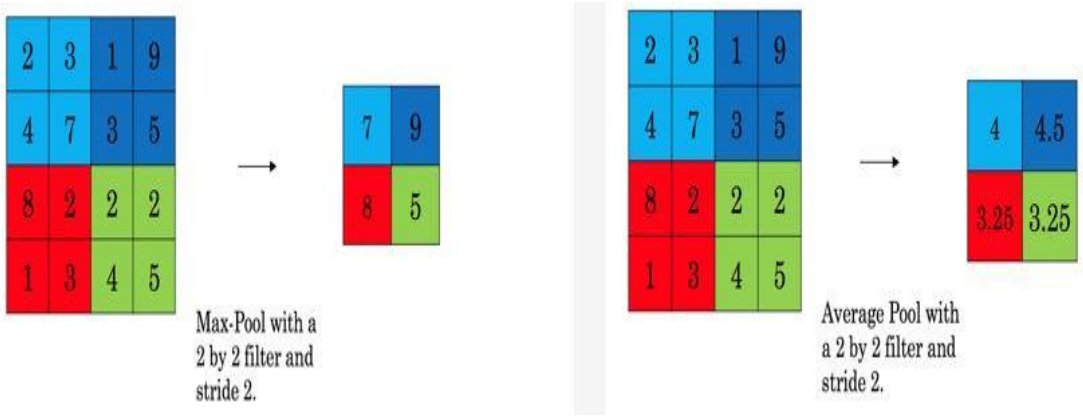

Figure 3.9: Max-pooling and average pooling

3.2.1.3. Fully connected layer

The fully contacted layer by contains three layers such as the flatting layer, the fully connected layers and finally the activation layer in our case the SoftMax classifier. I combined layers and used then as one layer for simplicity. each of these three have their own purpose as described below

3.2.1.3.1. Flatting layer

Up to know the outputs of the convolution layer or the outputs of the convolution layers are a 2 dimensional or a multiple dimensional volume to feed these outputs into the neural network these outputs must be converted in to a one-dimensional vector. The process of converting those volume outputs into a single vector is called flatten. As we have seen from the Figure below the flatten step takes 128x6x2 output form the max pool layer and changes it to a one-dimensional vector that can be feed as input to the neural network as shown in the Figure 3.10 below.

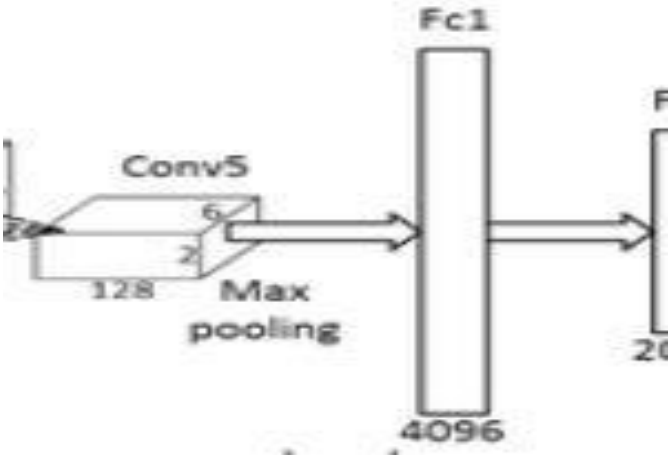

Figure 3.10: flattening a max pool output to a single vector

3.2.1.3.2 Fully connected layer

After the multi dimension volume output is flatten in to a single vector then these vectors are fully connected to first layer of the network and used as an input to the network the fully connected layer have various number of layers and various number of neurons on each layer depending on the problem on hand.

3.2.1.3.3SoftMax classifier

The finally layer of the convolutional neural network in our case is the SoftMax classifier these layers will output as the probability scores achieved by each class labels for the given input so we can use these outputs of the SoftMax classifier to classify the input as which level of class label it belongs to.

The outputs of the convolution layer are then fully connected to one or more layered Network and finally the SoftMax classification is performed.

CHAPTER 4

RESEARCH METHODOLOGY AND TOOLS

4.1. Research Methodology

The research methodology I have used the behavior analysis and play role analysis method because I am going to develop a model and tune (Play with) the parameters and hypermeters so as to get different models then I will analysis of their behavior those different models developed and finally get the best model that performs best on the development dataset and finally test these model on our test set. The objective of these study is to study the Ethiopian banknotes and develop a model to recognize and classify Ethiopian banknotes. For the development of the model, I have used is literature review and document analysis of different works performed directly image classification documents, papers written on Ethiopian bank notes and any papers written on other country banknotes.

Data set preparation is pre request for model development before you develop a model you must need the dataset you train ,develop and test the model so I am going to develop a brand new dataset which contains training dataset, development dataset and test dataset the percentages of the each datasets will contain and why will be explained in the chapter four which is experimentation then I select the appropriate neural network architecture and develop the model after I develop the model I will train the model using the training dataset, then tune the hypermeters of the model to get the best performance of the model on the development dataset then finally test the performance of the final model on the test set.

4.2. Data Collection

To collect the data from the above-mentioned sources I have used different techniques for the primary data I have collected the data using different devices the description of the devices is given on appendix 4. which have different image capturing ability in terms of image quality at different resolution or zooming level, different resolution and different backgrounds of the image and at deferent degree of rotation and the data we have collected have the image of banknotes is in one side of the banknote which is not the two sides of the banknotes.

4.2.1. Data sources

For these studies I have 2 data sources as a primary and secondary data source as a primary data I will take a number of different pictures of all of the Ethiopian banknotes so to take the pictures of all those banknotes I have to first collect the banknotes so I have collected different Ethiopian banknotes from all of my friends here and I got different distribution of banknotes I can like old, new, damaged ,not damaged and etc.. and since my primary data source is not enough, I will use my secondary data sources too.

4.2.1.1. Primary data sources

The major factor that affects the model from performing efficiently is the quality of the dataset. which means the performance of the model is highly dependent on the quality of the dataset we are going to use to train the model. To have high quality dataset we must need a data set for the distribution that we really want to apply the model it's like setting our target that we want to hit for example if we prepare the dataset from images got from the internet and train the model with that dataset and then we want to test the performance of the model on an image captured by users of the model using mobile images then the performance of the model will be really low because when we train the model the target we have to hit is images from internet which have high resolution ,framed and have high quality and we want to test the performance by an image captured by mobile devices which is low quality, highly noise, blurry, different backgrounds so we can't get good performance from the model with these test dataset so one thing we can do is make the dataset from the same distribution as possible and the image is going to be captured at different degree 45^o 30^o 15^o horizontal highly zoomed in , highly zoomed out, old banknotes, new banknotes, dirty banknotes and etc.

primary data for my datasets to be prepared, a good dataset I have to first collect the real bank notes with different nature such as old, new, broken where the dataset is going to be prepared so I asked all of my friends and collected all the banknotes possible some of the banknote pictures are shown in the Table 4.1 ,Table 4.2 and Table 4.3 Below then I take 2298 pictures for the frontside of the bank notes , 1200 pictures for the backside of the bank notes and 2298 pictures for the frontside of the bank notes at different orientation and degree and the banknotes that the image is prepared have different types like old banknote, new banknotes using 4 mobile

devices with different capturing ability and the number of pictures taken by those different capacity phones are given in the table below.

Table 4. 1: Number of data collection for the backside of bank notes

No	Notes Type	Camera	Total Pictures
1	5 ETB	5	387
2	10 ETB	5	387
3	50 ETB	5	387
4	100 ETB	5	387
Total			**1548**

Table 4. 2: Number of data collection for the frontside of bank notes

No	Notes Type	Camera	Total Pictures
1	5 ETB	5	553
2	10 ETB	5	405
3	50 ETB	5	263
4	100 ETB	5	326
Total			**1548**

Table 4.3: Number of data collection for the front and backside side bank notes

No	Notes Type	Camera	Total Pictures
1	5 ETB	5	572
2	10 ETB	5	572
3	50 ETB	5	593
4	100 ETB	5	561
Total			**2298**

The Ethiopian banknote of four denominations which I have collected from which the dataset is prepared the denominations are hundred (100), fifty (50) ETB, fifty (10) and one hundred (5) ETB which have shown on the Figures 4.1, Figure 4.2, Figure 4.3, Figure 4.4 below respectively.

Figure 4.1: One side view of Ethiopian 100 ETB banknote

Figure 4.2: One side view of Ethiopian 50 ETB banknote

Figure 4.3: One side view of Ethiopian 10 ETB banknote

Figure 4.4: One side view of Ethiopian 5 ETB banknote

4.2.1.2. Secondary data sources

As my secondary data source, I have used two techniques one is collect images of the banknotes which have high quality from the internet which makes our dataset distributed over different types of sources so increases the quality of the dataset and the second one is that there are different techniques that helps to increase the dataset that have been used without going to collect any dataset used when it's hard to get more data that enables you to increase your

dataset without going outside and collect pictures. These techniques are named as data augmentation the detail is given below.

a. **Data Augmentation**: - here data augmentation is mostly used when you have a network that overfits your data set and you need to regularize you can try other different techniques of regularizations like drop out but if those didn't work and increasing your data set is must and if it's very hard to get new fresh data then you have to use the data augmentation techniques described below. Here we since we are going to use ResNet with 50 layers and since we have around 5000 data it's obvious our network is going to overfit and I can't find other data so I have to use those techniques to increase my data set I will discuss how to use the augmentation in the implementation five. Technically data augmentation didn't only help us to increase our data set in my case it will help me to make the network learn different degree orientations of the image for example making the images rotate horizontally will help the network to learn and predict correctly if the user captures the banknote in reverse direction they also sometimes help us to learn the network different orientation or the actual capturing techniques that the user can may use

1. **Flip**: - these is flipping the image horizontally or vertically in our case we need we don't need flip since flipping the image didn't make sense when the banknote is rotated vertically.

Figure 4.5: The original version of Ethiopian banknote

Figure 4.6: The flipped version of 5 ETB banknote

2. **Rotation: -** rotation is rotating the image at some angel. Here rotating the image if the image is not square will affect the size of the image and this will help the network the learn images captured by the user at different angle than horizontal

Figure 4.7: The rotated version of 5 ETB banknote

3. **Scale: -** scale is the scaling of the image inward or outward which also helps the network to learn different scale images of the banknotes. The image you see below in Figure 3.8 is the scaled version of the original image of Figure 3.4 above. By scaled the width direction only by 256 we get the image below

Figure 4.8: Scaled version of the original 5ETB banknote

4. **Crop: -** crop the is randomly sampling some sections of the given image and reshaping the size to the size of the original image.

Figure 4.9: Cropped version of the 5 ETB banknote

5. **Gaussian Noise –** gaussian noise which has zero mean to essentially data points of all frequencies effectively distorted the high frequency features

Figure 4.10: Noised version of the original 5ETB banknote

4.3. Software Tools

As a software tool for these studies I am going to use Phyton programing language to develop the model, NumPy library to perform complex mathematical operations and Neural network framework which is TensorFlow framework to develop and train the Convolutional Neural Network model developed and some small programs are also written by me to preprocess the data like the program that perform those data augmentation described above, the network accepts the size of images the network will not accept different size of images so input images must have the same size but obviously the prepared data set will not have the same size so the dataset images must have to be made the same size by the program and the dataset is a collection of images but the network will not accept those image datasets they are must have to be converted to some readable form and those operation are also performed but the basic software's used at the top are described below techniques.

a. **Python**: - python is a high-level language for programing which is an object oriented and it is an integrated dynamic semantic primarily for app and web development developed by Guido Van Rossum in the 1991. Here for my model development I used Python as a programing language since python is the widely machine learning language for machine learning because most machine learning problems requires strong math skill and since python is easy to learn than other programing languages any mat skilled professional can learn the language and solve the machine learning problem.

b. **TensorFlow**: - is also an open source library developed by googles brain team for the purpose of numerical operations and it is an easy to use Application Programing Interface (API) that can help us implement machine learning algorithms and enable us to run efficiently both on Central Processing Unit (CPU) and Graphics Processing Unit (GPU).
c. **NumPy: -** is a python library that provides a support to perform calculations that are performed on large multidimensional arrays and calculations of elements of each complex matrix so that users can use the library to save development time and computational time.
d. **Keras:** -is also a high-level Neural Network specific Python API and it have the capability to run over other APIs or libraries like TensorFlow its main aim is to provide fast experimentation and it also allows you to prototyping fast and easily
e. **Jupyter Notebook: -** is an open source web-based application that consists of two categories such as the executable document or the computer program(code) written by programing languages like python and can be executed or run in order to perform a specific action and an easily human readable documents such as texts, paragraphs, heading, equations, graphs, links and etc.…
f. **Web Browser:** - to open the Jupyter notebook application I need a web browser the browser uses the localhost port to run on the local machine.

4.4. Data Analysis

The data that I 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.

CHAPTER 5

IMPLEMENTATION

5.1. Overview

the implementation phase doesn't only include the implementation of the model it also includes the implementation for the dataset which enables us to load the dataset from the hard disk and change it to h5 data format or some other reduced file format like CSV, TFrecords, tf.data, etc.… and making the data ready to be feed to the model.

The implemented model is implemented using python programming language using the kera's framework the model we are going to implemented is the ResNet the model has 50 layers grouped in to two groups such as the convolutional blocks and the identity blocks, we will discuss about those groups below. Finally, there is a SoftMax classifier with four class that categorizes the four Ethiopian banknote to their respective denominations.

5.2. Data Preparation

For the data that we used to train our model is a brand-new data set that I prepared for these purposes the dataset contains around 500 images which comprises of 400 images for each banknote denominations and which contains three categories the dataset that contains the front side of the bank notes which contains around 1500 images of row collected dataset and 100 images collected by using data augmentation techniques include images collected from the internet to make the dataset made of different distributions. The next group is the dataset that contains the backside of the banknotes which comprises around 1600 images from which 1300 images are images directly capture by me using four different devices with different image capturing quality and on different orientation of the banknotes and the 300 images are gathered by using data augmentation techniques including collecting image from internet. The third and final category contains 2200 images from which around 400 images are obtained by using data augmentation techniques like internet, flipping, scaling cropping and etc.

As I tried to mention above the primary focus of these dataset preparation is to make sure that the dataset is from the same distribution and includes different types of data as much as possible because if the dataset is not from the same distribution we miss our goal for example if we collected images only from the internet which have high quality and highly framed to learn the

model and test the model with pictures captured by mobile devices which are lower in quality, more blurry, noise then we can't get the best performance from the network because our target was to correctly classify banknote images with high-quality because we learn the model with high quality images the model will learn that but during the test time we give him a different data distribution so miss the goal so we can't get the best performance so making the training set ,development test and dataset from the same distribution is key thing we have to focus for these purpose we developed the data set with the same distribution with all the images such as high quality images, blurry images, noise images, zoomed-in, zoomed-out we have considered all the possible collections to our dataset and mix the all by focusing on our target which is to get best accuracy as possible on every category of the data so the collected have been from the same distribution.

The data format that we are using to store our dataset is H5 or HDF5 which stands for Hierarchical Data Format which saves a file in a hierarchical data format. Which stories a multidimensional array of a scientific data's and its commonly used in engineering, physics, aerospace, finance, academic research, medical fields. The most common versions of HDF file formats are HDF4 and HDF5 and we used the latest and efficient version of HDF which is HDF5 and HDF5 have a high ability of compressing(decreasing the dataset we have) for example my dataset have for both frontside and backside banknote dataset originally have 7.94GB images and know after it was changed to HDF5 file format the size of the dataset is reduced to MB which is around 237 MB and additionally its butter to access the HDF file from the disk rather than assessing the row images for the purpose of efficiency.

Then collected data have to be preprocessed such as reducing size, normalization and shuffling the data set because if one banknote denominals are in one order than after that the next banknote denominals then the network can't learn so to make the network learn the data must be shuffled in a random order then after the shuffling is performed to make the data ready to feed to the model put them to the training dataset but the images pixel values as they are can't be feed to the model they have to be converted to a NumPy array so the data set is converted to a NumPy array that makes the dataset easy to process the implementation of the dataset is given on the appendix four.

The hole dataset for each category is not going to be used as a tanning dataset instead the dataset of each category is going to be divided as training dataset, development dataset and test dataset

the implementation is shown in appendix 4b. The division of the dataset is performed by some common rule. The common rule is that if we have big dataset like in millions the division of the dataset to training, test and development is made by the ration 98%,1%,1% respectively or even less percent for the test and development datasets (Moindrot, O et al 2018) because here since we have big data 1% of a big dataset is enough for testing and development but if we don't have big dataset then the percentage for the training, test and development will be different it is like 60%,20%,20% respectively or 70%,15%,15% respectively theses is because when we have small dataset the test and development sets have also must be big enough to test the performance of the network and 20 or 15 % of a small dataset is not that big but 15or 20% of a millions dataset is to big.

So, for our dataset we don't have a million of data in our dataset we have only 2000 data in our dataset which is too small so the division of our dataset will have the proportion of 70% for training dataset, 15% for the test dataset and 15%for the development dataset.

Every time we run the model loading the images converting to NumPy array and processing it is not computationally efficient so we processed the data one which means we load the row images from the disk using the implementation which is given in appendix 4a, then we split the dataset to training dataset, test dataset and development dataset as specified above using the implementation given in the appendix 4b then I write these to the hard drive as HDF5 file format for an easy accesses in the feature when needed using the implementation given in appendix4.

5.3. Developing the Model

The developed model is a ResNet model which is a model with 50 layers. The model is programmed using python programming on a Keras framework which using the convolutional neural network the implementation of the model is shown on appendix three. Then the model will be trained using the training dataset the implementation of we use is on appendix four 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.

5.4. Model Architecture

The model that I have used to solve the problem is a ResNet which have two basic components which are the identity block and the convolutional block. Our model have around 50 layers and in theory of neural networks increasing the number of layers (making the network deeper) will always expected to increase the efficiency of the network but in practice it's difficult to train the model with big number of layers because of exploding gradient and sometimes vanishing gradients (K. He et al 2015) and to solve these problem there is a technique we use called the skip connection which allows us to take the activation from one layer and suddenly feed it to another layer even much deeper in the neural network which enables to build deeper neural networks without the problem of exploding or vanishing gradients by doing this I will build ResNet which enables us to learn very, very deep networks. ResNet are built out of something called residual block.

In today's neural network environment, the depth of neural networks has playing a crucial role in achieving a satisfying result (K. Simonyan et al ,2015) The architecture of the model is derived from the ResNet50 model. the model contains the identity layers, convolutional layer we combine those layers in different way and make the model. But before wee dive to the identity and convolution blocks of the layer I want to describe the problems of very deep networks.

Know a day's neural networks are becoming deeper , networks going from having just few layers like AlexNet (Krizhevsky,2017) to over hundreds of layers but the basic benefits of having vary deep neural networks is that it have the ability to represent vary complex functions, it have also the ability to learn features that are at different level of abstraction from edges from the (outer layer) to the complex features (From the deeper layer of the network) but there is a big problem to use these big networks and use them to learn a very complex features in our problem the problems that appear when we use very deep networks is that vanishing gradient and exploding gradient(Y.Bengio at al 2014 and X. Glorot et al 2015). Mostly vanishing gradient very deep neural networks often have a gradient that goes to zero the deeper we go the gradient goes to zero which makes the learning process too slow or almost zero these is because as you do back propagation from the final layer of the network back to the first layer of the network we are multiplying by the weight matrix at each steps and these makes the gradient to decrease exponentially quickly to zero which makes gradient descent unbearably slow as shown

in the Figure 5.1 below or in rare cases the gradient can may explode exponentially quickly and become a very large value. But these problems are solved normalized initialization (K. He et al 2015) and by using an intermediate normalization layer (S. Ioffe et al 2015) which helps the network with more layers in his network to start converging for stochastic gradient descent (SGD) with backpropagation but when deep networks begins to converge a new problem is introduced which is the degradation problem which is with the depth of the network increases the accuracy of the network gets saturated and then decreases but these problem comes not because of overfitting and adding more layers to an already suitably deep network increases the training error (R. K. Srivastava at al 2015 and K. He et al 2015). To solve this problem and enable you to build more deeper neural networks is to use the skip connection which allows you to take the activation from one layer and feed that in to another layer even much deeper in the network.

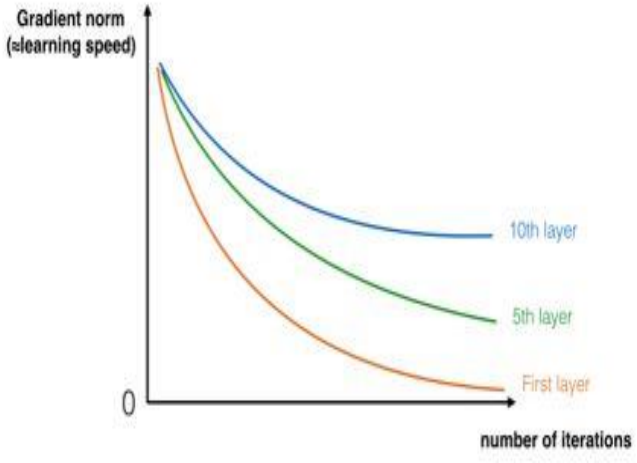

Figure 5.1: Learning rate saturation as the network becomes deep

5.4.1. Identity block

The blocks of any other convolutional networks are created by putting a number of layers which contains an activation a^l which is a^0 which equals to X (the input) for the first layer then a linear operation is performed and we got z^{l+1} then a ReLU or any nonlinearity operation is performed shown in the Figure 5.2 below so theses is performed in each layers of the network

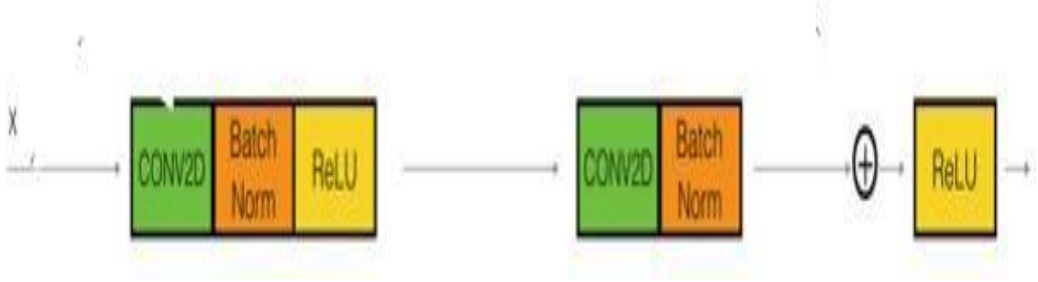

Figure 5.2: Normal path without skip connection

Identity block is the standard block used in ResNet but we take the activation from one-layer L say a^l take the activation and skip some layers and add that to another layers say L+2s activation say a^{l+2} which is deeper in the network. But here is one big problem which is the size of the activation we want to take and add say a^l will have different size than the activation that we are going to add say a^{l+2} to solve these we may use the same padding method that enables us to get the same output as the input after the convolution is performed. The implementation of our identity block is shown on appendix one.

And the identity block of the residual network is got when the input activation at some layer L say a^l have the same dimension as the activation we are going the add at say layer L+2 a^{l+2} here the addition operation is performed before the nonlinearly operation is performed as shown in Figure 5.3 below.

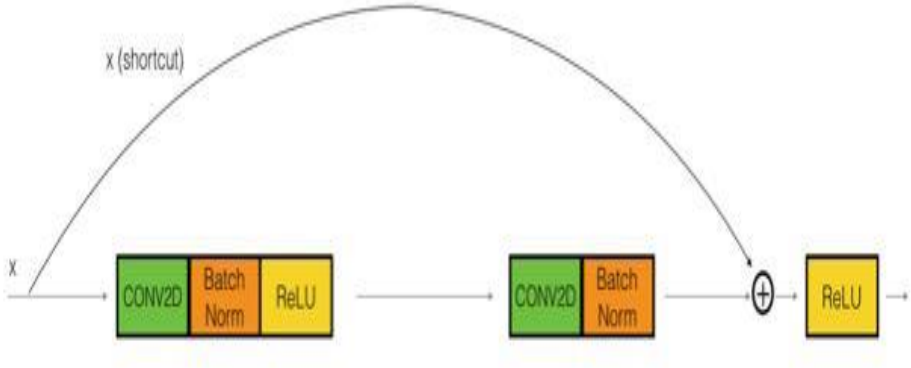

Figure 5.3: Skipping one layer in identity block

and the skip connection is not always skip one connection it can skip as many connection as its necessary the Figure 5.4 below shows skiping two layers.

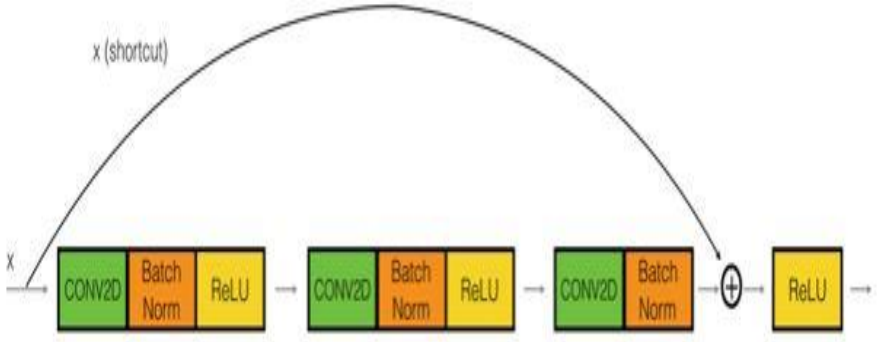

Figure 5.4: Skipping many layers in identity block

As we see from the graphs above there is batch normalization added after the convolution just before the ReLU operation to reduce the amount by which the hidden units will shift around what we call covariance shift so to we make the learning of the ResNet must faster and the ResNet with as much number of layers as possible is got by stacking a number of these blocks as much as we want to.

5.4.2. Convolutional block

The convolutional block is the other type of block where the input activation a^l which is a^0or maybe X if we are at the first layer have different dimension as the layer you want to add say a^{l+2} the basic difference with the identity block is that the convolutional block have an additional convolution layer in the shortcut path which helps us to resize the input a^l to a different size. So that the dimension of the input a^l will match up in the final addition needed to add to the shortcut value back to the main path as shown in the Figure 5.5 below. The entire implementation of the convolution layer was performed using keras frame work using python programing and the hole implementation was found in appendix two.

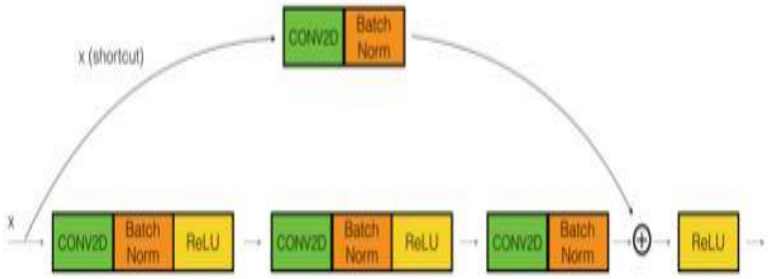

Figure 5.5: The convolutional block

The model I have developed is the combination of these models you can get the code for those blocks and the model in the appendix our model has 50 layers total I want to divide the models into five stages of implementation and describe them in detail so in the first stage I included I made 64 2D convolutions with the size of 7x7 and I used a 2x2 stride to move the 7x7 convolution on the input images pixel vertically and horizontally then a max pooling with a window of 3x3 with the stride of 2x2 the output of these layer is then feed to the next in the next stage contains three set of convolutions with 64,64,256 with a filter size of 3x3 and a stride of 1 and I used two identity blocks that uses three set of filters of size 64,,64,256 with a filter size of 3x3 in the third stage I again used a three 128,128,256 filters with a filter size of 3x3 and a stride of 2x2 again and I used three identity blocks that uses three set of filters of size 128,128,512 with a filter size of 3x3 in the fourth stage I used 256,256,1024 convolutions blocks with a filter size of 3x3 with a stride of 2x2 and I used five identity blocks that uses three set of filters of size 256,256,1024 with a filter size of 3x3 and in the fifth block I used a convolutional block that uses three set of filters of size 512,512,2048 with a filter size of 3x3 and a stride of 2x2 then I used two identity blocks that uses three identity filters of size 512,512,2048 with a filter size of 3x3 and finally a 2d average pooling is performed then the final output was flatten and feed to the fully connected(dense)layer which reduces the inputs flatten input to the number of class we want to classify using the SoftMax activation as shown in the Figure 5.6 below

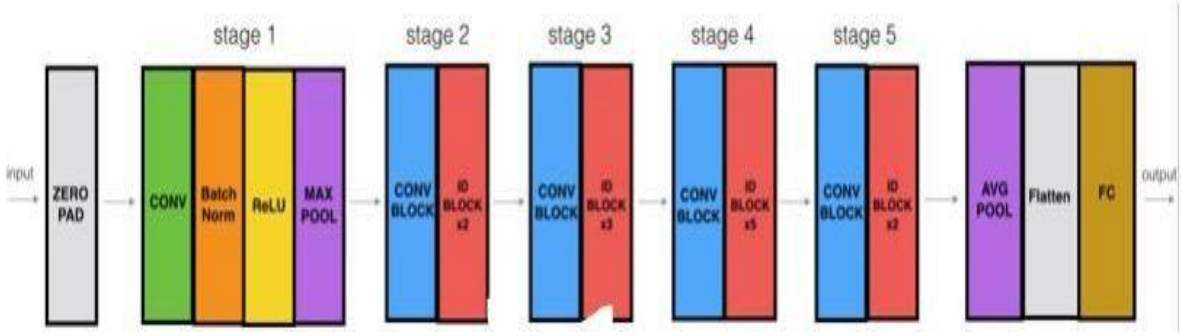

Figure 5.6: The hole design of the model

As you see from the hole models' picture Figure 5.6 above the CONV stands for the convolution blocks and ID BLOCK is for the identity blocks and CONVxn mean n convolution blocks and ID BLOCKxn means n number of identity blocks there are also max-pooling and batch normalizations in the model.

After the model have 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 the following parameters described below.

a. **Batch size: -**now we can able to train the developed model on our ready entire training dataset and here training on a large dataset is just slow (Better Mini-Batch Algorithms ,2019). For a gradient descent on our entire training set what we have to do is we have to process our entire training set before we take on a little step of gradient descent and we have to process the entire training set of say 2000 in our case or it may be millions in others training examples again before we take another little step of gradient descent so it turns out that we can get a faster algorithm if we let gradient descent starts to make some progress even before we finish processing the entire training set of hundred thousand or million training examples. In particular what we have to do to get an early progress is we split the training set into smaller little baby training sets. These baby training sets are called minibatches have their own sizes what we call batch size and these batch size have value from 1 up to N where N-is the number of training examples and these batch sizes define the number of examples in which we start to update the weight values of the model during training. So, in our case we are going to use minibatch

gradient descent because we want to make our algorithm run faster and since we have little dataset which is 2000, we are going to use 32 batch size.

b. **Image Size: -** the images in the dataset is captured with different devices so obviously they have different sizes (different Hight and weight) but to train the model we need to make the size of the images all similar. Here what we have to be careful is that if we down size the image too much it may difficult the identify the images if for human ayes and it will be harder for the model to learn quickly as an examples the images we original captured have a size of 2448x3264,3456x4608 and 1960x4032 so we can't feed these different and large size images directly to the model as they are so we reshaped the images to reshaped the images to 128x256. So, all the images will be changed to 128 by 256 images to be feed to the network during training, development, test or even during prediction time.

c. **Number Epoch** :- epoch is the move through the entire training set when we move once we say one epoch for example in our case we have 2000 dataset size and when we use 70% of our dataset for training dataset the number of training examples in the training set will be 1400 examples and size we have 32 batch size to move through the entire set with 32 batch size we have to iterate 44 times when we iterates 44 times and finish the hole training set we call it one epoch so to get better performance from our model we have to train the model many times as enough so in our case we trained the model 25 times which means we trained the model by moving through the entire dataset 25 times which makes the epoch size to be 32.

d. Learning rate: - is the amount by which the weight is updated during each iteration of gradient descent because $W = W - \alpha * dW$ so the value of 2 determines how fast will we learn. but there is no one value for learning rate that can be used in every situation it depends on a lot of values its one of the hypermeters(reference) that we have to tune during the training of the model.

e. Optimizer: - is an algorithm that will be used in order to make gradient descent so that to enable the network to learn. the optimizer that we are going to use is Adam optimizer. I chose Adam optimizer because it has the combined befits of both AdaGrad (Duchi et al., 2011) and RMSProp (Tieleman & Hinton, 2012)

(DiederikP, & Jimmy, 2015) and it has also a lot of advantages (DiederikP et al 2015) such as.

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

CHAPTER 6

RESULT AND DISCUSSION

6.1. Result

The result we obtained is this study is basically the model that can perform well in the training and test datasets in terms of these we have achieved a great result. I have developed model which uses ResNet as a base. the model is trained in three different datasets such as the frontside banknotes dataset, the backside of the banknote's dataset and the combination of both the backside and the front side banknote datasets and give us three different models İ have assessed the performance of those three models and I have achieved a great performance in all the three models.

The performances of those models are measured in terms of the loss value they have and the accuracy value they have achieved

Loss: - is a metrics to evaluate the performance of a model it's a calculated value which indicates that how bas our model is in predicting on a single example. In measuring a model using loss if we want to make our model to perform well, we have to decrease the loss value as much as possible. The goal here is to find the weight and the bias values that will give us the smallest loss value which leads to high performance of the model.

Accuracy: - is also an evaluation mastic which is used to evaluate the performance of a given model. It can be calculated as the number of predictions that our model predicted correctly divided by the total number of predictions that our model has made.

So, based on these loss and accuracy performance measurement metrices I have measured the performance of my three models. Let's see them one by one.

For the first category of the dataset with 70/15/15 dataset splitting in the first model trained for the front sides of the banknote the loss value will become 0.0091 the accuracy value in my case have two values such as the accuracy of the model on the training dataset and the accuracy value of the model on the test dataset so the result obtained by the first model is 99.73% accuracy on the training dataset and 96.20% on the test dataset as shown in the figure below on Figure 6.1 and Figure6.2 and described in the Table 6.1 shown below.

train the model

```
5]: Front_model.fit(X_train, Y_train, epochs = 10, batch_size = 32)

Epoch 1/10
1105/1105 [==============================] - 2296s 2s/step - loss: 0.0390 - acc: 0.9873
Epoch 2/10
1105/1105 [==============================] - 2254s 2s/step - loss: 0.0258 - acc: 0.9973
Epoch 3/10
1105/1105 [==============================] - 2262s 2s/step - loss: 0.0262 - acc: 0.9910
Epoch 4/10
1105/1105 [==============================] - 2254s 2s/step - loss: 0.0497 - acc: 0.9846
Epoch 5/10
1105/1105 [==============================] - 2251s 2s/step - loss: 0.0243 - acc: 0.9928
Epoch 6/10
1105/1105 [==============================] - 2252s 2s/step - loss: 0.0193 - acc: 0.9928
Epoch 7/10
1105/1105 [==============================] - 2255s 2s/step - loss: 0.0038 - acc: 1.0000
Epoch 8/10
1105/1105 [==============================] - 2239s 2s/step - loss: 0.0491 - acc: 0.9955
Epoch 9/10
1105/1105 [==============================] - 2237s 2s/step - loss: 0.0384 - acc: 0.9891
Epoch 10/10
1105/1105 [==============================] - 2243s 2s/step - loss: 0.0091 - acc: 0.9973
```

Figure 6.1: training performance at the last epochs of first model

```
# Test Performace

In [7]: preds = Front_model.evaluate(X_test, Y_test)
        print ("Loss = " + str(preds[0]))
        print ("Test Accuracy = " + str(preds[1]))

        316/316 [==============================] - 207s 655ms/step
        Loss = 0.1589392082694965
        Test Accuracy = 0.9620253179646745
```

Figure 6.2: test performance at the last epochs of first model

For the first category of the dataset with 70/15/15 dataset splitting in the second model we have obtained almost the same result as the first model which is 0.0259 loss value and the training and test dataset accuracy obtained is 99.08% and 98.38% accuracy respectively as shown below in Figure 6.3 and Figure 6.4 and described in the Table 6.1 shown below.

train the model

```
In [5]: Backside_model.fit(X_train, Y_train, epochs = 6, batch_size = 32)

Epoch 1/6
1082/1082 [==============================] - 2206s 2s/step - loss: 0.0572 - acc: 0.9926
Epoch 2/6
1082/1082 [==============================] - 2179s 2s/step - loss: 0.0022 - acc: 1.0000
Epoch 3/6
1082/1082 [==============================] - 2181s 2s/step - loss: 0.0294 - acc: 0.9908
Epoch 4/6
1082/1082 [==============================] - 2189s 2s/step - loss: 0.2281 - acc: 0.9464
Epoch 5/6
1082/1082 [==============================] - 2185s 2s/step - loss: 0.0991 - acc: 0.9713
Epoch 6/6
1082/1082 [==============================] - 2183s 2s/step - loss: 0.0259 - acc: 0.9908
```

Figure 6.3: training performance at the last epochs of second model

```
# performance on Test dataset

In [6]: preds = Backside_model.evaluate(X_test, Y_test)
        print ("Loss = " + str(preds[0]))
        print ("Test Accuracy = " + str(preds[1]))

        310/310 [==============================] - 204s 657ms/step
        Loss = 0.0568769050701972
        Test Accuracy = 0.9838709665882972
```

Figure 6.4: test performance at the last epochs of second model

For the first category of the dataset with 70/15/15 dataset splitting in the third model which is trained on the combination of back and front banknotes I have achieved a good performance which is a loss value of 0.0359% and accuracy of 98.82%on the training dataset and an accuracy of 96.086956% accuracy on the test dataset as shown below on Figure 6.5 and Figure 6.6 and described in the Table 6.1 Shown below.

train the model

```
In [24]: model_allside_note_9925_9413.fit(X_train, Y_train, epochs = 2, batch_size = 32)

Epoch 1/2
1608/1608 [==============================] - 3377s 2s/step - loss: 0.0631 - acc: 0.9801
Epoch 2/2
1608/1608 [==============================] - 3850s 2s/step - loss: 0.0359 - acc: 0.9882
```

Figure 6.5: training performance at the last epochs of third model

```
# test Perormance

In [25]: preds = model_allside_note.evaluate(X_test, Y_test)
         print ("Loss = " + str(preds[0]))
         print ("Test Accuracy = " + str(preds[1]))

         460/460 [==============================] - 308s 670ms/step
         Loss = 0.15884092151308837
         Test Accuracy = 0.9608695652173913
```

Figure 6.6: training performance at the last epochs of third model

For the first category of the dataset with 80/10/10 dataset splitting in the first model trained on the front sides of the banknote the loss value will become 0.0012%. The accuracy value in my case have two values such as the accuracy of the model on the training dataset so the result obtained by the first model is 100 % accuracy on the training dataset and 97.419% on the test dataset as shown below on Figure 6.7 and Figure 6.8.

```
In [5]: Front_model.fit(X_train, Y_train, epochs = 5, batch_size = 32)

        Epoch 1/5
        1237/1237 [==============================] - 2553s 2s/step - loss: 0.0012 - acc: 1.0000
        Epoch 2/5
        1237/1237 [==============================] - 2513s 2s/step - loss: 0.0017 - acc: 0.9992
        Epoch 3/5
        1237/1237 [==============================] - 2522s 2s/step - loss: 0.0012 - acc: 1.0000
        Epoch 4/5
        1237/1237 [==============================] - 2514s 2s/step - loss: 7.1337e-04 - acc: 1.0000
        Epoch 5/5
        1237/1237 [==============================] - 2526s 2s/step - loss: 0.0012 - acc: 1.0000

Out[5]: <keras.callbacks.History at 0x2f122039128>
```

Figure 6.7: training performance at the last epochs of front model

Evaluating the model on the test set

```
preds = Back_model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
155/155 [==============================] - 101s 651ms/step
Loss = 0.06402163404610849
Test Accuracy = 0.9741935522325578
```

Figure 6.8: test performance of front model

In the Second model trained on the back sides of the banknote the loss value will become 0.0060%. The accuracy value in my case have two values such as the accuracy of the model on the training dataset so the result obtained by the first model is 99.68% accuracy on the training dataset and 99.32% on the test dataset as shown below on Figure 6.9 and Figure 6.10.

train the model

```
In [5]: Back_model.fit(X_train, Y_train, epochs = 10, batch_size = 32)
Epoch 2/10
1237/1237 [==============================] - 2497s 2s/step - loss: 0.0492 - acc: 0.9854
Epoch 3/10
1237/1237 [==============================] - 2502s 2s/step - loss: 0.0763 - acc: 0.9741
Epoch 4/10
1237/1237 [==============================] - 2504s 2s/step - loss: 0.0203 - acc: 0.9935
Epoch 5/10
1237/1237 [==============================] - 2500s 2s/step - loss: 0.0136 - acc: 0.9935
Epoch 6/10
1237/1237 [==============================] - 2497s 2s/step - loss: 0.0132 - acc: 0.9951
Epoch 7/10
1237/1237 [==============================] - 2502s 2s/step - loss: 0.0470 - acc: 0.9846
Epoch 8/10
1237/1237 [==============================] - 2504s 2s/step - loss: 0.0246 - acc: 0.9943
Epoch 9/10
1237/1237 [==============================] - 2502s 2s/step - loss: 0.0086 - acc: 0.9976
Epoch 10/10
1237/1237 [==============================] - 2501s 2s/step - loss: 0.0060 - acc: 0.9968
Out[5]: <keras.callbacks.History at 0x16170b5deb8>
```

Figure 6.9: training performance at the last epochs of back model

Evaluating the model on the test set

```
[5]: preds = Back_model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
149/149 [==============================] - 97s 654ms/step
Loss = 0.03925152816153413
Test Accuracy = 0.9932885906040269
```

Figure 6.10: test performance of the back model

In the third model trained on the all sides of the banknote the loss value will become 0.0017%. The accuracy value in my case have two values such as the accuracy of the model on the training dataset so the result obtained by the first model is 100% accuracy on the training dataset and 100% on the test dataset as shown below on Figure 6.11 and Figure 6.12. and as summarized in the Table 5.1 below.

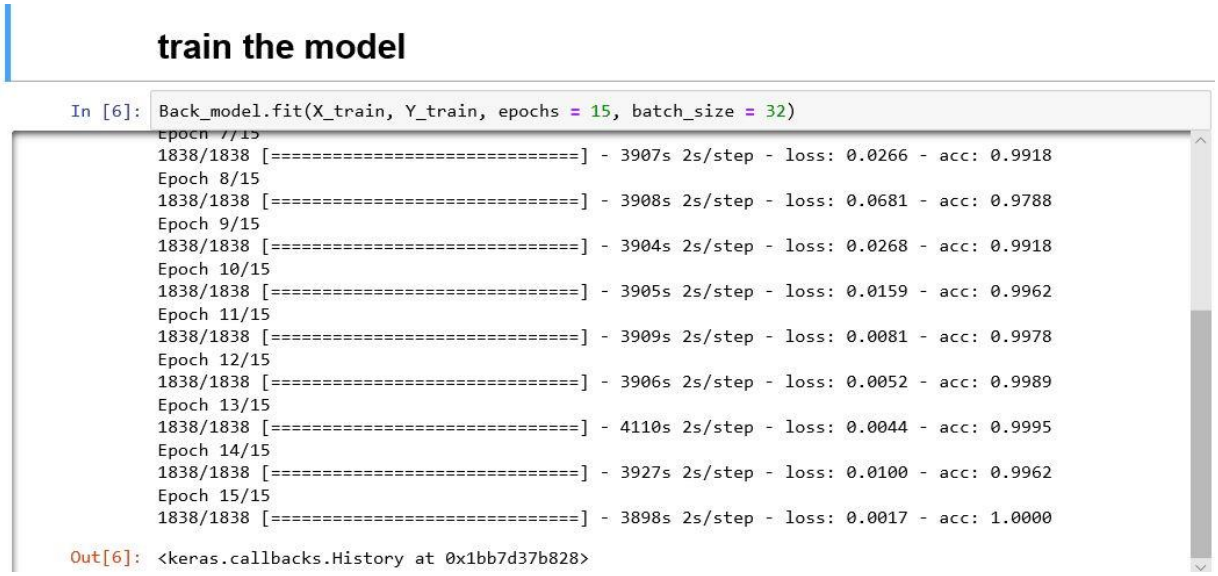

Figure 6.11: training performance at the last epochs of all side model

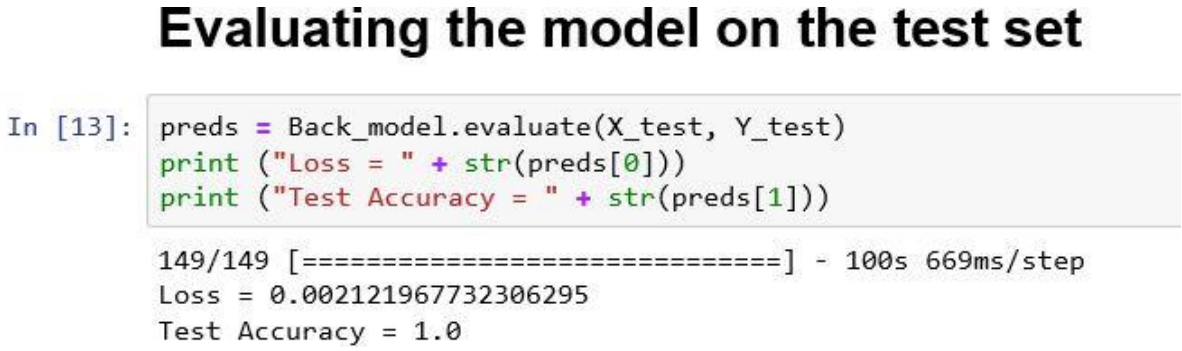

Figure 6.12: test performance of all side model

Table 6.1: summery 70/15/15 dataset split performances on each model

No	Model name	Loss value in %	Training accuracy in %	Test Accuracy in %
1	Front model	0.0091	99.73	96.20
2	Back model	0.0259	99.08	98.38
3	All side model	0.0359	98.82	96.086956

Table 6.2: summery 80/10/10 dataset split performances on each model

No	Model name	Loss value in %	Training accuracy in %	Test Accuracy in %
1	Front model	0.0124	100	97.41
2	Back model	0.0060	99.68	99.32
3	All side model	0.0017	100	100

From above Table 6.1 and Table 6.2 we have seen that the performance of the models by splitting the dataset with 70/15/15 and 80/10/10 for training, development and test respectively, as we see from the table on average when use the 80/10/10 proportion of the training, development and test respectively we obtained high performance on the loss value, training accuracy value and test accuracy value, so an increase in the training dataset increased the performance of the model. specially on the test set since we have decreased the test dataset by 5% the number of images in the test have decreased but the performance obtained on the test have increased on all the three model. On the other side the 70/15/15 have 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 are smaller on the loss value, on the training accuracy and on the test 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.

6.2 Discussion

The model performances for all models is obtained by training the model for more than 100 epochs. And achieving high performance on the training or loss doesn't guaranty high performance on the test data set when I train the model more I got 99.25% accuracy on the training set but I got 94.13% accuracy on the test set these shows that achieving high value on the training set doesn't mean that aching high performance on the test dataset so I chose the model with 98.13% accuracy on the training set and have 96.304 accuracy on the test dataset because the our goal we want to achieve high is our test dataset.

From the above the loss I loss or accuracy I obtained is not obtained in one gradient descent the model has performed a number of gradient descents in every batch which is 32 in each epoch. For each model we have performed different number of epochs for example the number of epochs performed for the first model to obtain the specifies result we have performed 40 epochs. And the model we got is obtained by tuning the hyperparameter such as the learning rate, number of layers, number of neurons in each layer, learning rate decay value and etc.

From the model's performance I understand that the performance in terms of the accuracy metrics or in terms of the loss metrics doesn't always decrease for the loss or doesn't always decrease for the accuracy their values always increases and decreases which means it oscillates in the process of moving the loss to the minimal point and the loss to the point of maximal point

The Figure 1 to Figure 11 above shows the values of the performance measurements used and they are obtained by tuning the hypermeters of the model we have tuned so by tuning the models we have obtained different models of each categories of our problem to do so İ have used the development dataset after that I have selected the best models for the three categories of my problems such as classifying the bank notes using the frontside of the banknotes' ,the classifying the bank notes using the backsides of the banknote and finally classifying the bank notes using both the front and backsides of the banknotes together the hyperparameter we have used for those best performing selected models are described in the table below.

6.2.1. Human level performance

Human level performance is used to know whether the model can need further improvements or not the concept is that a given model can perform at list at a human level or a little more to a human level. A model can't perform much more than a human level is a model performs at exactly human level the model didn't need any further improvements. But if the model performs much less than human performance the model needs an improvement so in our case the human level accuracy is 100% because humans can easily classify the Ethiopian banknotes to their corresponding denomination easily without any error.

As we have seen above the models is so exciting because the models perform nearly at a human level but here we have to know that the dataset is not prepared from images scanned by high quality IMR scanning devices instead the dataset images are obtained by capturing images by using any ordinary mobile devices but the result we achieved is similar to the result of other

studies that are made by using high quality images. These shows that Neural Networks can perform better than any other techniques of banknote classification even when the images or the datasets do not have high quality.

but the result of these study is also to develop a brand new dataset of the Ethiopian banknotes in terms of the dataset we have developed three brand new datasets such as the dataset that contain the front sides of the banknotes, the second one which contains the back sides of the banknotes and finally the dataset that contains the combination of both the front side and back side of the banknotes all of them are made of images with different distributions. The banknotes are collected from customers which the banknotes are available in most of its forms so the dataset we prepared can enable us to hit our target because we have used banknotes which are different natures such as old, new, damaged and medium damaged and we captured the images using four different mobile devices which makes our dataset to be a distributed dataset which includes all possibilities available.

CHAPTER 7

CONCLUSION AND RECOMMENDATION

7.1 Conclusion

To measure weather your model needs further improvement or not one thing I can do is the identify what is the human level performance of that given problem which mean at what accuracy value does a human can perform to the same given problem with the same dataset that we used by the model. so that we can compare that value with the human performance value if the model performance is equal or better than the human performance the model didn't need any more improvement but if the human level performance is better than the model performance then the model needs more improvement from this I understand that the human level performance for identifying Ethiopian banknotes on my dataset is 100% so the performance of the model is almost equal to the human level performance.

Since we have used data sets that are made up of images that are captured by any ordinary mobile device which have low image quality but the result we obtained is similar to those other techniques that use complicated computations and images that are captured by so expensive high quality devices the result İ have obtained show that using neural networks for image classification is much more butter than the other techniques.

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

The performance of the model is highly determined by a number of 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.

7.2. Recommendation

even if our results are very near to the human level performance but if the performance is not equal to or greater than the human level performance there is still a place to improvement so feature study is needed to make the model performs at exactly equal as a human level or to make them perform better than human level.

During the training process of the model we trained the model for the three separate datasets as we have discussed in chapter five but the training time takes a lot of time because of the size of the image that the model takes as input is high so I highly recommend to others to don't develop the model from scratch instead takes already trained models that have similar architecture as yours and if their architecture is exactly similar architecture as yours take the trained model and train the trained model using your own dataset or if the architecture is not exactly similar take the weight and bias values of the already trained model and use them as an initial weight and bias values of your model.

The Keras frame work where İ have made the model is so suitable for developing mobile applications that uses machine learning so for other researchers I highly recommend to use these models and or dataset and develop a mobile application for these problems.

REFERENCES

Bala N., and Rani, U., Emerg ,J., and Manag,R. (2014). Paper currency recognition, 2014; 3:77–81.

Bengio, P., Simard., and Frasconi, P. (2014). Learninglong-termdependencies with gradient descent is difficult., 5(2):157–166, 2014.

Berhane, Y., Worku, A., Bejiga, A., Adamu, L., Alemayehu, W., Bedri, A., and Kebede, T. (2008). National Survey on Blindness, Low Vision and Trachoma in Ethiopia: Methods and Study Clusters Profile. *Ethiopian Journal of Health Development*, 21(3), 1-4. doi:10.4314/ejhd. v21i3.10049.

Bhurke, C., Sirdeshmukh, M., and Kanitkar, M. (2015). Currency recognition using image processing. *Int. J. Innov. Res. Comput. Commun. Eng*. 2015;3:4418–4422.

Szegedy, W., Liu, Y., Jia, P., Sermanet, S., Reed, D., Anguelov, D., Erhan, V., and Rabinovich, A. (2015). Going deeper with convolutions. *In Conference on Computer Vision and Pattern Recognition (CVPR), 2015.*

Diederik, P., and Diederik, P. (2015). Adam : a method for sthocastic optimization 2015.

Dunai, L., Pérez, M., Peris, F., and Lengua, I. (2017). Euro Banknote Recognition System for Blind People. Sensors, 17(12), 184. doi:10.3390/s17010184

Fentahun, H. (2014). Automatic recognition of Ethiopian paper currency, 2014 from page 11-16

Gai, S., Yang, G., and Wan, M. (2013). Employing quaternion wavelet transform for banknote classification.Neurocomputing. *2013; 118:171–178. doi: 10.1016/j.neucom.2013.02.029.*

Glorot, X., and Bengio, Y. (2015). Understanding the difficulty of training deep feedforward neural networks. *In International Conference on Artificial Intelligence and Statistics (AISTATS), 2015.*

Hasanuzzaman, F., Yang, X., and Tian, Y. (2012). Robust and effective component-based banknote recognition for the blind. *IEEE Trans. Syst. Man Cybern. Part C.*

He, X., Zhang, S., and Sun., j. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In International Conference on Computer Vision (ICCV,*) 2015.

Ioffe,S., and Szegedy, C. (2015). Batch normalization: Accelerating deep networktrainingbyreducinginternalcovariateshift. *In International Conference on Machine Learning (ICML)*,2015.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2017-05-24). "ImageNet classification with deep convolutional neural networks" (PDF). Communications of the *Association for Computing Machinery (ACM).* 60 (6): 84–90. doi:10.1145/3065386. ISSN 0001-0782.

Kwon, S., Pham T., Park, K., Yoon S., and Jeong, D. (2016). Recognition of banknote fitness based on a fuzzy system using visible light reflection and near-infrared light

Lee, J., Hong, H., Kim, K., and Park, K. (2017). A survey on banknote recognition methods by various sensors. Sensors. 2017; 17:313 doi: 10.3390/s17020313.

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.

Moindrot, O., & Genthial, G. (2018, January 24). Splitting into train, dev and test sets. Retrieved March 12, 2019, from *https://cs230-stanford.github.io/train-dev-test-split.html*

Tuyen, D., Pham,D., Eun, Lee., and Ryoung, P. (July 2017). Multi-National Banknote Classification Based on Visible-light Line Sensor and Convolutional Neural Network. (2017). Sensors, 17(7), 1595. doi:10.3390/s17071595

Pawade, D., Chaudhari, P., and Sonkambale H. (2013). Comparative study of different paper currency and coin currency recognition method. Int, 2013.

Pham, T., Park Y., Kwon S., Park K., Jeong, D., and Yoon, S. (2016). Efficient banknote recognition based on selection of discriminative regions with one-dimensional visible-light line sensor.

Rahman, M., Poon, B., Amin, M., and Yan, H. (2014). Recognizing Bangladeshi Currency for Visually Impaired. Communications in Computer and Information Science Machine Learning and Cybernetics.

Raimond, Z., and Dinku, K. (2009). counterfeit currency identification system - a case study on. *Journal of EEA, 26, 74-78.*

Rashid, A., Prati, A., and Cucchiara, R. (2013). On the design of embedded solutions to banknote recognition.

Safraz, M. (2015). An intelligent paper currency recognition system. Proc. *Comput. Sci. 2015*;

Seung, Y., Tuyen, D., Kang, R., Dae, S., and Sungsoo, Y. (2016). transmission images. Sensors.

Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *In International Conference on Learning Representations (ICLR)*, 2015.

Sekeroglu, B., Khashman, A. (2005). Multi-banknote identification using a single neural network; *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems; Antwerp,* Belgium. 20–23 September 2005.

Srivastava, K., and Schmidhuber, j. (2015). Highway networks. arXiv:1505.00387, 2015.

Tessfaw, M., Ramani, M., and Bahiru, M. (2018). Ethiopian Banknote Recognition and Fake Detection Using Support Vector Machine. *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT).*

Viola, P., and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features; *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition; Kauai, HI, USA. 8–14 December 2001*.

Youn, S., Choi, E., Baek, Y., and Lee, C (2015). Efficient multi-currency classification of CIS banknotes. Neurocomputing. 2015.

Zeggeye, J., and Assabie, Y. (2016). Automatic Recognition and Counterfeit Detection of Ethiopian Paper Currency. *International Journal of Image, Graphics and Signal Processing,* 2016.02.04

APPENDICES

Appendices 1

Identity Block Implementation

```
def identity_block(X, f, filters, stage, block):
    """
    Implementation of the identity block
    Arguments:
    X -- input tensor of shape (m, n_H_prev, n_W_prev, n_C_prev)
    f -- integer, specifying the shape of the middle CONV's window for the main path
    filters -- python list of integers, defining the number of filters in the CONV layers of the main path
    stage -- integer, used to name the layers, depending on their position in the network
    block -- string/character, used to name the layers, depending on their position in the network
    Returns:
    X -- output of the identity block, tensor of shape (n_H, n_W, n_C)
    """
    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'
    # Retrieve Filters
    F1, F2, F3 = filters


    # Save the input value. You'll need this later to add back to the main path.
    X_shortcut = X
```

```
    # First component of main path
    X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1), padding = 'valid', name = conv_name_base + '2a', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)
    ### START CODE HERE ###
    # Second component of main path (≈3 lines)
    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1), padding = 'same', name = conv_name_base + '2b', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)
    # Third component of main path (≈2 lines)
    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1), padding = 'valid', name = conv_name_base + '2c', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)
    # Final step: Add shortcut value to main path, and pass it through a RELU activation (≈2 lines)
    X = layers.Add()([X_shortcut, X])
    X = Activation('relu')(X)
    ### END CODE HERE ###
    return X
```

Appendices 2

Convolution Block Implementation

```
def convolutional_block(X, f, filters, stage, block, s = 2):
    """
    Implementation of the convolutional block
    Arguments:
    X -- input tensor of shape (m, n_H_prev, n_W_prev, n_C_prev)
    f -- integer, specifying the shape of the middle CONV's window for the main path
    filters -- python list of integers, defining the number of filters in the CONV layers of the main path
    stage -- integer, used to name the layers, depending on their position in the network
    block -- string/character, used to name the layers, depending on their position in the network
    s -- Integer, specifying the stride to be used
    Returns:
    X -- output of the convolutional block, tensor of shape (n_H, n_W, n_C)
    """
    # defining name basis
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'
    # Retrieve Filters
    F1, F2, F3 = filters
    # Save the input value
    X_shortcut = X
    ##### MAIN PATH #####
```

```
    # First component of main path
    X = Conv2D(F1, (1, 1), strides = (s,s), padding = 'valid',name = conv_name_base + '2a', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)
    ### START CODE HERE ###
    # Second component of main path (≈3 lines)
    X = Conv2D(F2, (f, f), strides = (1,1), padding = 'same',name = conv_name_base + '2b', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)
    # Third component of main path (≈2 lines)
    X = Conv2D(F3, (1, 1), strides = (1,1), padding = 'valid',name = conv_name_base + '2c', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)
    ##### SHORTCUT PATH #### (≈2 lines)
    X_shortcut = Conv2D(F3, (1, 1), strides = (s,s),padding = 'valid', name = conv_name_base + '1', kernel_initializer = glorot_uniform(seed=0))(X_shortcut)
    X_shortcut = BatchNormalization(axis = 3, name = bn_name_base + '1')(X_shortcut)
    # Final step: Add shortcut value to main path, and pass it through a RELU activation (≈2 lines)
    X = layers.Add()([X_shortcut, X])
    X = Activation('relu')(X)
    ### END CODE HERE ###
    return X
```

Appendices 3

Model Implementation

```
def ResNet50(input_shape = (128, 256, 3), classes = 4):
    """
    Implementation of the popular ResNet50 the following architecture:
    CONV2D -> BATCHNORM -> RELU -> MAXPOOL -> CONVBLOCK -> IDBLOCK*2 -> CONVBLOCK -> IDBLOCK*3
    -> CONVBLOCK -> IDBLOCK*5 -> CONVBLOCK -> IDBLOCK*2 -> AVGPOOL -> TOPLAYER
    Arguments:
    input_shape -- shape of the images of the dataset
    classes -- integer, number of classes
    Returns:
    model -- a Model() instance in Keras
    """
    # Define the input as a tensor with shape input_shape
    X_input = Input(input_shape)
    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input)

    # Stage 1
    X = Conv2D(64, (7, 7), strides = (2, 2), name = 'conv1', kernel_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = 'bn_conv1')(X)
```

```
X = Activation('relu')(X)

X = MaxPooling2D((3, 3), strides=(2, 2))(X)


# Stage 2

X = convolutional_block(X, f = 3, filters = [64, 64, 256], stage = 2, block='a', s = 1)

X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')

X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')


### START CODE HERE ###


# Stage 3 (≈4 lines)

X = convolutional_block(X, f = 3, filters = [128, 128, 256], stage = 3, block='a', s = 2)

X = identity_block(X, 3, [64, 64, 256], stage=3, block='b')

X = identity_block(X, 3, [64, 64, 256], stage=3, block='c')

X = identity_block(X, 3, [64, 64, 256], stage=3, block='d')


# Stage 4 (≈6 lines)

X = convolutional_block(X, f = 3, filters = [256, 256, 1024], stage = 4, block='a', s = 2)

X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')

X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')

X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')

X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')

X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')
```

```
    # Stage 5 (≈3 lines)
    X = convolutional_block(X, f = 3, filters = [512, 512, 2048], stage = 5, block='a', s = 2)
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='b')
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')
    # AVGPOOL (≈1 line). Use "X = AveragePooling2D(...)(X)"
    X = AveragePooling2D(pool_size=(2, 2),name='avg_pool')(X)
    ### END CODE HERE ###
    # output layer
    X = Flatten()(X)
    X = Dense(classes, activation='softmax', name='fc' + str(classes), kernel_initializer = glorot_uniform(seed=0))(X)
#End of the model
    # Create model
    model = Model(inputs = X_input, outputs = X, name='ResNet50')
    return model
```

Appendices 4

Implementation for the Dataset Preparation

a. Loading the images to the program

```
taning_data_set=[]
datadir="images/"
categories=["Data Set"]
classes=["5","10","50","100"]
def create_traning_data():
    for category in categories:
        path=os.path.join(datadir,category)
        for folders in classes:
            finalpath=os.path.join(path,folders)
            class_num=classes.index(folders)
            print(finalpath)
            print(class_num)
            for img in os.listdir(finalpath):
                try:
                    img_array=cv2.imread(os.path.join(finalpath,img))
                    new_array=cv2.resize(img_array,(256,128))
                    taning_data_set.append([new_array,class_num])
                except Exception as e:
                    pass


create_traning_data()
print('completed')
```

b. Splitting the data to training set , test set and development set

```
def createsplited_dataset(taning_data_set):
    train_X=[]
    train_Y=[]
```

```
test_X=[]
test_Y=[]
dev_X=[]
dev_Y=[]

split_1 = int(0.7 * len(taning_data_set))
split_2 = int(0.9 * len(taning_data_set))

train_filenames = taning_data_set[:split_1]
dev_filenames = taning_data_set[split_1:split_2]
test_filenames = taning_data_set[split_1:split_2]

for features,label in train_filenames:
   train_X.append(features)
   train_Y.append(label)
for features,label in test_filenames:
   test_X.append(features)
   test_Y.append(label)
for features,label in dev_filenames:
   dev_X.append(features)
   dev_Y.append(label)

train_X=np.array(train_X).reshape(-1,128,256,3)
train_Y=np.array(train_Y).reshape(-1,1)

test_X=np.array(test_X).reshape(-1,128,256,3)
test_Y=np.array(test_Y).reshape(-1,1)

dev_X=np.array(dev_X).reshape(-1,128,256,3)
dev_Y=np.array(dev_Y).reshape(-1,1)
return train_X,train_Y,test_X,test_Y,dev_X,dev_Y
```

c. Writing the dataset to hard drive as HDF5 file format

```
def write_dataset():

    train_X,train_Y,test_X,test_Y,dev_X,dev_Y=createsplited_dataset(taning_data_set)

    pickle_ut=open('train_set_x.pickel','wb')
    pickle.dump(train_X,pickle_ut)
    pickle_ut.close()
    pickle_ut=open('train_set_y.pickel','wb')
    pickle.dump(train_Y,pickle_ut)
    pickle_ut.close()

    pickle_ut=open('dev_X.pickel','wb')
    pickle.dump(dev_X,pickle_ut)
    pickle_ut.close()
    pickle_ut=open('dev_Y.pickel','wb')
    pickle.dump(dev_Y,pickle_ut)
    pickle_ut.close()

    pickle_ut=open('test_set_x.pickel','wb')
    pickle.dump(test_X,pickle_ut)
    pickle_ut.close()
    pickle_ut=open('test_set_y.pickel','wb')
    pickle.dump(test_Y,pickle_ut)
    pickle_ut.close()

    hf = h5py.File('datasets/train_data_allside_signs.h5','w')
    hf.create_dataset('train_set_x',data=train_X)
    hf.create_dataset('train_set_y',data=train_Y)
    hf.close()
```

```
    hf = h5py.File('datasets/test_data_allside_signs.h5','w')
    hf.create_dataset('test_set_x',data=test_X)
    hf.create_dataset('test_set_y',data=test_Y)
    hf.close()

    hf = h5py.File('datasets/dev_data_allside_signs.h5','w')
    hf.create_dataset('dev_set_x',data=dev_X)
    hf.create_dataset('dev_set_y',data=dev_Y)
    hf.close()
    return 'sucessesfuly writen'
```

d. Reading the prepared dataset to the program

```
train_dataset = h5py.File('datasets/train_data_allside_signs.h5', "r")
    train_set_x_orig = np.array(train_dataset["train_set_x"][:]) # your train set features
    train_set_y_orig = np.array(train_dataset["train_set_y"][:]) # your train set labels

    test_dataset = h5py.File('datasets/test_data_allside_signs.h5', "r")
    test_set_x_orig = np.array(test_dataset["test_set_x"][:]) # your test set features
    test_set_y_orig = np.array(test_dataset["test_set_y"][:]) # your test set labels

    dev_dataset = h5py.File('datasets/dev_data_allside_signs.h5', "r")
    dev_set_x_orig = np.array(dev_dataset["dev_set_x"][:]) # your test set features
    dev_set_y_orig = np.array(dev_dataset["dev_set_y"][:]) # your test set labels

    train_set_y_orig = train_set_y_orig.reshape((1, train_set_y_orig.shape[0]))
    test_set_y_orig = test_set_y_orig.reshape((1, test_set_y_orig.shape[0]))
    dev_set_y_orig = dev_set_y_orig.reshape((1, dev_set_y_orig.shape[0]))

    return train_set_x_orig, train_set_y_orig, test_set_x_orig, test_set_y_orig,
dev_set_x_orig,dev_set_y_orig
```

Appendices 5

Used Camera Specifications

no	Images size	Hight pixel	Width pixel	Dimension
1	12MP	2248	3264	2248x3264
2	18.5mp	2224	1080	2224x3264
3	8mp	1960	4032	1960x4032
4	12	2448	3264	2448x3264
5	16mp	3456	4608	3456x4608