

**RECOGNIZING AND UNDERSTANDING HANDWRITTEN JOINT LETTER
IN BANGLA**



DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh
December 2019

**RECOGNIZING AND UNDERSTANDING HANDWRITTEN JOINT LETTER
IN BANGLA**

BY

**TAPSARA AKTER
ID: 161-15-681**

This Report Presented in Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Supervised By

Md. Nazmul Hoq
Lecturer
Department of CSE
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 2019

APPROVAL

This Project titled “**RECOGNIZING AND UNDERSTANDING HANDWRITTEN JOINT LETTER IN BANGLA**”, submitted by Tapsara Akter (ID:161-15-681) to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 26 November, 2019.

BOARD OF EXAMINERS

.....

Dr. Syed Akhter Hossain

Chairman

Professor and Head

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

.....

Dr. S M Aminul Haque

Internal Examiner

Associate Professor & Associate Head

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

.....

Saif Mahmud Parvez

Internal Examiner

Lecturer

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

.....

Dr. Mohammad Shorif Uddin

External Examiner

Professor

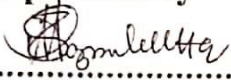
Department of Computer Science and Engineering

Jahangirnagar University

DECLARATION

I hereby declare that this project has been done by me under the supervision of **Md. Nazmul Hoq, Lecturer, Department of CSE Daffodil International University**. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of a degree or diploma.

Supervised by:

 4/12/19
.....

Md. Nazmul Hoq

Lecturer

Department of CSE

Daffodil International University

Submitted by:

Tapsara
.....

Tapsara Akter

ID: 161-15-681

Department of CSE

Daffodil International University

ACKNOWLEDGEMENT

First, I would like to express my heartiest thanks and appreciation to almighty God for His divine blessing that made me capable to finish the last year project successfully.

I am extremely thankful and wish my profound indebtedness to **Md. Nazmul Hoq, Lecturer, Department of CSE**, Daffodil International University, Dhaka. Deep Knowledge & keen interest of my supervisor in the field of Machine learning to complete this project. His direct supervision, academic direction, nonstop support, thoughtful suggestions, helpful analysis, continual encouragement, reading many inferior drafts and correcting them at every stage have made it possible to finish this project.

I would like to express my heartiest gratitude to **Prof. Dr. Syed Akhter Hossain, Head of Department of CSE**, for his kind help to finish my project and also to other faculty members and the staff of CSE department of Daffodil International University.

I would like to thank my entire course mate in Daffodil International University, who took part in this discussion while completing the course work.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

ABSTRACT

Recently, some work has been done on Bangla Handwriting character recognition. But many of them are only capable of recognizing Bangla digits, vowel or consonant character only. For building a model that can recognize Bangla handwriting, it is important to work with joint letter also. This “Recognizing & understanding handwritten joint letter in Bangla” is a project to develop a model to recognize and understand Bangla handwritten joint letters. This model will be capable of taking single or multiple images as input and then it will automatically recognize which joint letter is in the image. For this project I have collected the dataset from multiple sources.

In this project, I have used transfer learning with deep learning for getting best result. Here I implemented some Deep Neural Network (DNN) and compared the results of them. Among all of them ResNet50 provides the best result. The result of the model showed 96.37% accuracy on testing dataset which is better than other existing works.

Keywords — Convolutional Neural Network, Transfer Learning, Handwritten Joint Letter Recognition, Handwritten Joint Letters.

TABLE OF CONTENTS

CONTENT	PAGE
Approval	ii
Board of examiners	iii
Declaration	iv
Acknowledgements	v
Abstract	vi
List of Tables	ix
List of Figures	x
Chapter 1: Introduction	1-4
1.1 Introduction	1
1.2 Motivation	2
1.3 Rationale of the Study	3
1.4 Research Questions	3
1.5 Expected Output	3
1.6 Report Layout	4
Chapter 2: Background	5-10
2.1 Introduction	5
2.2 Related Works	7
2.3 Research Summary	9
2.4 Scope of the Problem	9
2.5 Challenges	9
Chapter 3: Research Methodology	11-30
3.1 Introduction	11
3.2 Research Subject and Instrumentation	11
3.3 Data Collection Procedure	27
3.4 Statistical Analysis	27
3.5 Implementation Requirements	28

CONTENT	PAGE
Chapter 4: Experimental Results and Discussion	30-36
4.1 Introduction	30
4.2 Experimental Results	30
4.3 Descriptive Analysis	35
4.4 Summary	36
Chapter 5: Summary and Conclusion	37-38
5.1 Summary of the Study	37
5.2 Conclusions	37
5.3 Recommendations	38
5.4 Limitation	38
Appendices	39
Appendix A: Research Reflection	39
Appendix B: Related Issues	39
References	40-41
Plagiarism Result	42-44

LIST OF TABLES

TABLES	PAGE
Table 2.1.1 Types of joint letter present in database	5
Table 3.4.1 Statistical analysis	27
Table 4.2.1 Experimental result of other architectures	31
Table 4.3.1 Comparison with other works	36

LIST OF FIGURES

FIGURES	PAGE
Figure 1.1.1 Example of Bangla joint letter	2
Figure 3.2.1 Block diagram of methodology	12
Figure 3.2.2 Architecture of CNN	13
Figure 3.2.3 Architecture of VGG16	15
Figure 3.2.4 Architecture of ResNet50	17
Figure 3.2.5 Architecture of Inception	20
Figure 3.2.6 Architecture of MobileNet	22
Figure 3.2.7 Architecture of DenseNet	24
Figure 3.2.8 Home page of the project	25
Figure 3.2.9 Joint letter page of the project	25
Figure 3.2.10 Matching page of the project	26
Figure 3.2.11 Image of the project database	26
Figure 4.2.1 Confusion matrix	30
Figure 4.2.2 Graph of VGG16	32
Figure 4.2.3 Graph of InceptionV3	32
Figure 4.2.4 Graph of Xception	33
Figure 4.2.5 Graph of MobileNet	33
Figure 4.2.6 Graph of DenseNet121	34
Figure 4.2.7 Graph of ResNet50	34
Figure 4.3.1 Testing accuracy for architectures	35

CHAPTER 1

INTRODUCTION

1.1 Introduction

Bangla is the national and official language in Bangladesh and the seventh most spoken language in the world [1]. As it is such a popular language, there is a wide area for research. Many research works are available to recognize English language and some works are available on Bangla handwriting recognition. Most of the works are done for bangla vowel , consonant and bangla digit recognition. A few works are done on bangla joint letter recognition.

In this project I am working on Bangla joint letter recognition. Because without building a model that can recognize joint letter correctly, it's not possible to recognizing bangla language completely. This project will take input image and then it will automatically provide the output of which joint letter is it. This work will also play an important role for further research works. This project will be helpful for all people to reading Bangla ID cards, Name plates, License plate recognition, Post office automation etc.

Recognizing handwriting faces a big challenge that is enormous variety of handwriting pattern by different people. Figure 1.1.1 is an example of some Bangla joint letter.

ক	খ	গ	ঘ	ঙ	চ	ছ
জ	ট	ঠ	ড	ঢ	ণ	
ত	থ	দ	ধ	ন	প	ফ
ব						

Figure 1.1.1 Example of Bangla Joint Letters

1.2 Motivation

Now-a-days so many research works has been done on handwritten Bangla letter recognition. But still rare researchers are attempting to recognize Bangla handwritten joint letter.

- This project will help to recognize and understand handwritten Bangla joint letter. In this project a model will be proposed that can recognize and understand Banga joint letter more accurately than previous works.
- This project will be beneficiaries for all people when they need to convert Bangla to english from any handwritten image document. Because this part will help the system to recognize bangla joint letter that will help the system to understand Bangla language more effectively.
- Helpful for understanding Handwritten document in Bangla.

1.3 Rationale of the Study

A lot of work has been done on English language recognition. Compared with English language, there has not been so many research work of Bangla language on the handwritten letter recognition and understanding. Several research works is done on the handwritten Bangla vowel, consonant, digit recognition. But there is hardly found any work that has done on handwritten Bangla joint letter recognition. So now-a-days it becomes important to work on Bangla language recognition.

The goal of this project is, training Deep Neural Network with labeled images of joint letters. Using this network the model will be capable to recognize the image of joint letters. This approach is used to develop software that will get the ability of understanding Bangla joint letter more accurately by using a proposed model that will get the best accuracy.

1.4 Research Questions

1. What are the way of recognizing Bangla Handwritten joint letter?

How to recognize Bangla handwritten joint letter by using Deep Neural Networks (DNNs)?
And what are the process?

2. How to measure Accuracy of the proposed model?

How to implement Deep Neural Network (DNN) models and which model provides the best accuracy for testing data?

1.5 Expected Output

In this project, single or multiple images can be given as input in proposed model. And model will automatically identify the category of joint letter for each of those images. That means, people can get output at a time for single or multiple images. But that image should contain single joint letter.

1.6 Report Layout

This report is organized as follows:

Chapter 2 will consist of the discussion about the background of the research like the related works, research summary, and challenges. In chapter 3 we will discuss about the methodology of work. On Chapter 4 discussion will be continued about experimental results of this research experiment. Chapter 5 will be discussed about the summary, conclusion, limitation of the research.

CHAPTER 2

BACKGROUND

2.1 Introduction

In this project I am working with bangla handwritten joint letter recognition. When two or more consonant letters are joined as a single letter than it's called joint letter. up to four consonant letter can be represented as a single joint letter. In bangla language almost 285 types of joint letters are available [2]. 172 types of joint letter are available in my database that I collected for my project.

Table 2.1.1 contains those 172 types of joint letters:

Table 2.1.1 types of joint letter present in database

pa_rafal a(প্র)	sa_ta a(স্ব)	sa_b a(স্ব)	na_t, a(ন্ট)	na_d ,a(ন্ড)	s'a_c a(শ্চ)	ka_s a(স্ব)	na_s a(স্ব)	pa_p a(প্প)	da_b a(দ্বব)	pa_l a(প্প)	ga_n a(গ্ন)	ma_ma(শ্ম)
n''a_ja(ঞ)	na_t a_uk ar(ক্ত)	ta_ta (ত)	ka_r afala (ক্র)	pha_ rafal a(ফ্র)	t,a_r afala (ট্র)	s'a_r afala (শ্র)	ma_ bha(ষ)	ba_l a(ব্ল)	t,a_t, a(ট্ট)	sa_ta _rafa la(স্র)	ja_b a(জ্বব)	na_ta_ra fala(ব্র)
ka_s,a(স্ব)	ga_r afala (গ্র)	ka_t a(ক্ত)	na_n a(ন্ন)	ma_ pa(ম্প)	ta_b a(ত্ব)	da_r afala (দ্র)	s'a_ ba(স্ব)	pa_t a(প্ত)	ga_l a(গ্ল)	ca_c a(চ্চ)	s,a_k a(স্ক)	ma_pa_r afala(ম্প্র)
sa_t,a_r afala(স্ট্র)	sa_t, ha(স্বহ)	na_t, a(ন্ট)	da_d a(দ্দ)	sa_k a(স্ক)	la_la (ল্ল)	sa_p a(স্প)	ba_d a(ব্দ)	ka_l a(ক্ল)	la_t, a(ল্ট)	sa_m a(স্ম)	ta_m a(ত্ম)	sa_t,a_ra fala(স্ট্র)
na_d,a_r afala(ন্ড্র)	sa_t, a(স্ট)	la_p a(ল্প)	na_d ha(ন্ডহ)	ja_n' 'a(ঝঞ)	ba_r afala (ব্র)	n'a_ ga(গ্ন)	s'a_ na(স্ন)	ta_ra fala(ত্র)	s,a_t, h(স্টহ)	pa_n a(প্পন)	d,a_r afala (দ্র)	s,a_ka_r afala(স্ক্র)
ca_cha(চ্ছ)	ma_ ba(ম্ব)	s,a_t a(স্ট)	n'a_ ka(স্ক)	ka_t, a(ক্ট)	n''_c a(স্ক)	ma_ pha(ম্পহ)	ka_ ma(ক্ম)	da_g a(দগ)	na_d a(ন্ড)	t,a_b a(ট্টব)	ca_n ''a(চ্ছ)	sa_pha(স্পহ)

na_ta(ত্ত)	ka_k a(ক)	sa_ta _uka r(সু)	na_ ma(ন্ম)	d,a_ ga(ড গ)	ba_j a(জ)	ha_r afala (হ)	pa_s a(প)	s,a_p ha(ফ)	s'_c ha(শ ছ)	ja_jh a(জা)	da_d a(দ)	n'a_ka_s ,a(ন'ক্ষ)
na_da_r afala(ন্দ)	na_h a(নহ)	dha_ ba(ধ ব)	gha_ rafal a(ঘ)	ja_ja _ja(জ ব)	sa_n a(স)	pa_r afla_ ukar(প ফ)	la_b a(ল ব)	ma_ na(ম)	gha_ na(গ)	ga_g a(গ)	ka_s, a_na (ক্ষ ন)	ga_rafal a_ukar(গ ফ)
ta_ta_ba (ত্তব)	d,a_ d,a(ড ড)	ha_n a(হ)	da_b ha(ড)	s'a_ ma(স ম)	na_d a_ba (ন্দ ব)	ta_ra fla_u kar(ত ফ)	ca_c ha_b a(চ ব)	n,a_ n,a(ন)	la_m a(ল)	sa_ra fala(স ফ)	ma_l a(ম)	s,a_pa_r afala(স প)
ja_ja(জ)	ba_b a(ব)	bha_ rafal a(ভ)	s'a_l a(স)	da_ ma(দ)	ha_ ma(হ)	bha_ la(ভ)	ma_ bha_ rafal a(ম ভ ফ)	s,a_p a(স প)	n'a_ kha(ন খ)	ja_ra fala(জ ফ)	ga_b a(গ ব)	da_rafal _ukar(দ ফ)
ka_s,a_ ma(ক্ষ ম)	na_t, a_raf ala(ন ত ফ)	la_k a(ল ক)	ba_d ha(ব দ)	ka_b a(ক ব)	n'a_ gha(ন খ)	s'a_r afla_ ukar(স ফ ফ)	n.a_ ba(ন ব)	ha_b a(হ ব)	da_d a_ba (দ ব)	da_g ha(দ গ)	dha_ rafal a(ধ ফ)	n,a_dha _rafala(ন খ ফ)
s,a_na(স ফ)	na_t, ha(ন ত)	sa_la (স ল)	s,a_ ma(স ম)	ma_r afala (ম ফ)	kha_ rafal a(খ ফ)	ha_l a(হ ল)	ga_ ma(গ ম)	la_g a(ল গ)	sa_k ha(স ক)	t,ha_ rafal a(ত হ ফ)	pha_ la(প ল)	sa_ka_ra fala(স ক ফ র)
na_ba(ন ব)	pa_t, a(প ত)	ta_n a(ত ন)	t,ha_ ba(ত হ ব)	ga_d ha(গ দ)	ta_t, ha(ত ত)	n'a_ ma(ন ম)	la_d a(ল দ)	n,a_ ma(ন ম)	n''a_ cha(ন খ)			

This project is a small work of machine learning (ML). Machine learning focuses on making predictions using computer systems to perform a specific task without relying on pattern or explicit instruction. Using dataset machine learning algorithms build model based on training data from that dataset. Machine learning algorithms are used for filtering email, computer vision etc.

Deep learning is a part of machine learning method based on artificial neural networks (ANNs). Deep learning can be supervised, unsupervised and semi-supervised. Deep neural network has a

most commonly applied class, which is convolutional neural network (CNN). Convolutional Neural Network use convolution in place of general matrix multiplication in minimum one of their layers.

Transfer learning is an experimental problem in machine learning that while solve any problem it stores the knowledge gained from solving that problem and make use of that gained knowledge for solving different but related problem.

2.2 Related Works

At present, many research works are done on Bangla language recognition. Those works are done mainly on bangla digit recognition, vowel and consonant recognition. Some of those research work achieved good accuracy rate.

In the paper [3], they are worked with Boise state Bangla handwriting dataset. Which contains 104 word or 364 character essay that uses 49 basic characters, all 11 vowels and 32 consonants. With this dataset and 3 other dataset they obtained classification accuracy 96.8% with SVM classifier based on a cubic kernel. This dataset help to develop isolated character, number, consonant conjuncts, characters with diacritics recognition, line identification, word spotting etc. demographic information capable to help research into handwriting variations based on age group, sex, profession and left/right handedness.

In the paper [4], they use Hidden Markov Model (HMM) to recognize handwritten Bangla words. They proposed a method that break up works in zone-wise and then perform HMM based recognition. This approach makes the work easy. They use a dataset of 10120 Bangla handwritten words. They combine LGH feature with HMM that helps to recognize the words in zone-wise. And SVM is used for the identification of modifiers. In their experiment they actually

showed that zone based segmentation have strong recognition capability. They used 7725 images for training and 2395 for testing.

In the paper [5], they present a technique for Bangla/English script identification to the destination address block of Bangladesh envelopes images. This technique is only capable of identifying Bangla/English scripts on the real Bangladesh postal images. Weak part of this experiment is, relatively simple technique can reach a high accuracy level for discriminating among English script. In their experiment the proposed algorithm typically classified into four categories: (a) the schemes based on analysis of connected components. (b) the schemes based on analysis of characters, words and text lines. (c) the schemes based on analysis of text blocks. (d) the schemes based on analysis of hybrid information of connected components, text lines etc. The accuracy of script identification is very high for printed text, and for handwritten text, the proposed approach achieved a satisfactory of accuracy 95%.

In the paper [6], they use CNN model for classifying Bangla Handwriting Character that contains 11 vowels and 39 consonants Bangla characters. For their proposed model they gets 98%, 96.81%, 95.71% and 96.40% validation accuracy for CMATERdb, IST, BanglaLekha. Isolated dataset and mixed dataset respectively. Proposed model was trained with one dataset and cross-validation with two other datasets. Limitation of this model is it become confused to understand overwritten character and incorrect labeling of some images that dataset contained. This mode performed poorly if the train on noise-free data.

In the paper [7], they discussed on a set of the state-of-the-art-deep convolutional neural network (DCNNs) and their performance. DCNN can extract discriminative features from raw data and represent them with a high degree of invariance to object distortion. For DenseNet they get recognition rate of 99.13% for handwritten Bangla digits, 98.31% for handwritten Bangla alphabet and 98.18% for special character recognition.

In the paper [8], a process is proposed that recognize and convert images of individual Bangla handwritten characters into electronically editable format. With 200000 images, the accuracy of the test set is 95.25% using Convolutional Neural Network (CNN).

2.3 Research Summary

In this report, I am going to find a method to recognize handwritten joint letter. There is a lot of variety in people's handwriting style. Sometimes understanding some of them are very difficult. Now because of the abundant variety of handwriting style, this becomes very tough for me to choose a model that can recognize handwriting. For this reason I have implemented

- Sequential Convolutional Neural Network (CNN)
- And some architectures of CNN. Which are MobileNet, ResNet50, DenseNet121, VGG16, InceptionV3, Xception.
- ResNet50 perform best.
- Transfer learning applied model is better than sequential model.

2.4 Scope of the Problem

1. Using this project people can provide input image to find the result of which joint letter is in the input image and the input should be a single joint letter.
2. In this project I have worked on single joint letter recognition. This project can't identify joint letter from any paragraph.
3. This project will only applicable for Bangla joint letter recognition.

2.5 Challenges

1. Problem in fixing image size for resizing the image.
2. Selecting proper classification methods.
3. Implementing those classification methods in python. Because it took a lot of time for running models.

CHAPTER 3

Research Methodology

3.1 Introduction

Handwriting recognition is difficult because of the enormous variety of handwriting styles of different people. In this project, I tried to build a model by trying some Deep Neural Network (DNN) such as sequential convolution Neural Network (CNN), some architecture of CNN like ResNet50, Xception, Inception v3, VGG16, denseNet121, MobileNet. After implementing this algorithms I will compare the results.

And also trying to develop software that will take image input, and then it will recognize the input image of the joint letter by performing one of algorithms that work best.

3.2 Research Subject and Instrumentation

Handwriting recognition is difficult because of the enormous variety of handwriting styles of different people. In this project, I have tried to build a model that provides better result of recognition. For this, I implemented Convolutional Neural Network (CNN) and some other architectures of CNN such as ResNet50, Xception, InceptionV3, VGG16, DenseNet121, MobileNet and compare provided results among them. Full process has many phases which shown in Figure 3.2.1

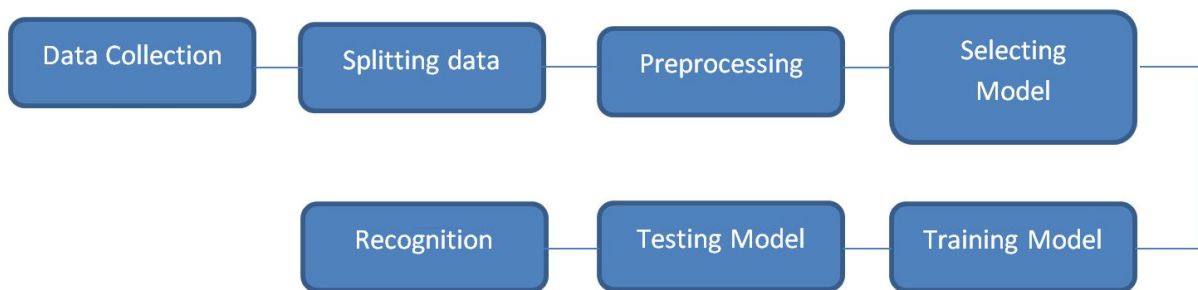


Figure 3.2.1 Block diagram of methodology

Splitting Data:

At first I split my joint letter dataset in test set and train set for testing and training model. For sequential Convolutional Neural Network (CNN) I split the dataset into 75% in train set and 25% in test set . And for other architecture splitting size was 80% in train set and 20% in test set.

Image preprocessing:

Preprocessing process is performed to get the proper result. For sequential CNN I resize all the selected image of our target dataset to 50*50 pixels. Then I change it from RGB to Gray image. Then I remove noise and add labeling to images. For other archichitecture of CNN I have used **ImageDataGenerator** function.

Selecting model, Training model, testing model:

I will implement Convolutional Neural Network (CNN) and some other architectures of CNN such as ResNet50, Xception, InceptionV3, VGG16, DenseNet121, MobileNet to get the perfect or better result.

Sequential Convolutional Neural Network (CNN):

- **Convolutional Neural Network (CNN)[9]:**

I will resize all images to 50*50 pixels. Converted all dataset's 50*50 pixel image into a 784+(1 label)d matrix and then store all the image pixel into a CSV file for fast computation. Now perform a minmax normalization to reduce the effect of illumination differences. Then split the data into test and train data. Add a densely-connected layer

with 64 unites to the model and then add another one. A softmax is added with 10 output units. Then train the model with batch size 50 and epochs 15. Testing the model and evaluate the model by calculating the test accuracy which is 82.37 %. And then draw the confusion matrix.

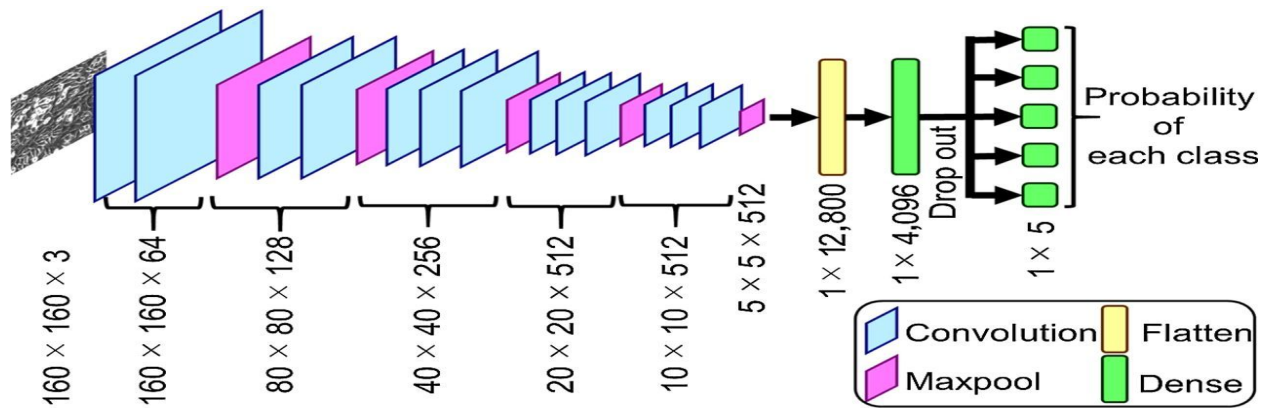


Figure 3.2.2 architecture of CNN

Architectures of Convolutional Neural Network (CNN):

- **VGG16 [10]:**

VGG16 is a convolution neural net (CNN) architecture. To implementing this architecture, at first I imported all the necessary libraries. Like **KERAS**, **OS**, **NUMPY** which I will need to implement VGG16. I also imported **ImageDataGenerator** from **KERAS**. Then I need preprocessing to import data with labels easily into model. Then I import VGG16 architecture from **KERAS** application. Then I create an object of **ImageDataGenerator** for training and testing data and passing the folder which has train data to the object trdata and test data to the object tsdata. The **ImageDataGenerator** will automatically label all the data inside each folder with its same padding. After initializing the model I add

- 2x convolution layer of 64 channel of 3×3 **KERNEL** and use same padding
- 1x maxpool layer of 2×2 pool size and stride 2×2
- 2x convolution layer of 128 channel of 3×3 **KERNEL** and use same padding
- 1x maxpool layer of 2×2 pool size and stride 2×2

- 3x convolution layer of 256 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2

We also add Rectified Linear Unit (ReLU) activation to each layer. For which all the negative values are not passed to the next layer.

After creating the entire convolution I pass the data to the dense layer so for that I flatten the vector which comes out of the convolutions and add

- 1x dense layer of 4096 units
- 1x dense layer of 4096 units
- 1x dense Softmax layer of 10 units

After that my model is finally prepared. Now I need to compile the model.

I will use Adam optimizer to reach the global minima while training out model. Then I will specify the learning rate of the optimizer, here in this case it is set at 0.001. Now I can check the summary of the model by using `model.summary()` . After creating the model I will import ModelCheckpoint and EarlyStopping method from KERAS. I will create an object of both ModelCheckpoint and EarlyStopping and pass that as callback functions to `fit_generator`. After that the model will start to train and I will start to see the training/validation accuracy and loss. Now I can visualize training/testing accuracy and loss using matplotlib. For VGG16 the accuracy is 96.04%. Figure 3.2.3 shows the architecture of VGG16.

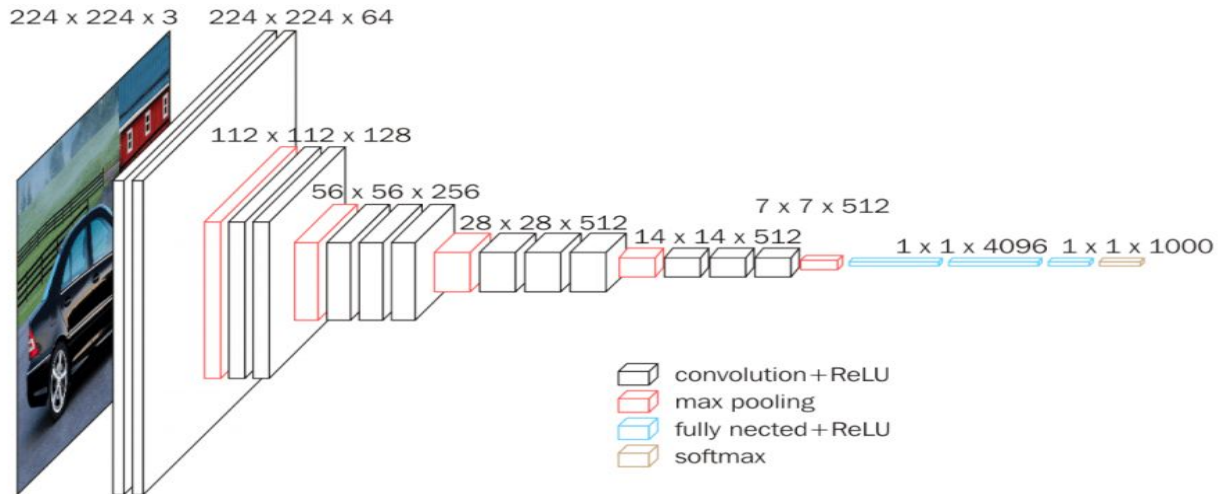


Figure 3.2.3 architecture of VGG16

- **ResNet50 [11]:**

ResNet50 is a convolution neural net (CNN) architecture. To implementing this architecture, at first I imported all the necessary libraries. Like **KERAS**, **OS**, **NUMPY** which I will need to implement ResNet50. I imported **ImageDataGenerator** from **KERAS**. Then I need preprocessing to import data with labels easily into model. Then I import ResNet50 architecture from **KERAS** application. Then I create an object of **ImageDataGenerator** for training and testing data and passing the folder which has train data to the object trdata and test data to the object tsdata. The **ImageDataGenerator** will automatically label all the data inside each folder with its same padding. After initializing the model I add

- 2x convolution layer of 64 channel of 3×3 **KERNEL** and use same padding
- 1x maxpool layer of 2×2 pool size and stride 2×2
- 2x convolution layer of 128 channel of 3×3 **KERNEL** and use same padding
- 1x maxpool layer of 2×2 pool size and stride 2×2
- 3x convolution layer of 256 channel of 3×3 **KERNEL** and use same padding
- 1x maxpool layer of 2×2 pool size and stride 2×2
- 3x convolution layer of 512 channel of 3×3 **KERNEL** and use same padding
- 1x maxpool layer of 2×2 pool size and stride 2×2

- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2

I also add Rectified Linear Unit (ReLU) activation to each layer. For which all the negative values are not passed to the next layer.

After creating the entire convolution I pass the data to the dense layer so for that I flatten the vector which comes out of the convolutions and add

- 1x dense layer of 4096 units
- 1x dense layer of 4096 units
- 1x dense Softmax layer of 10 units

After that my model is finally prepared. Now I need to compile the model.

I will use Adam optimizer to reach the global minima while training out model. Then I will specify the learning rate of the optimizer, here in this case it is set at 0.001. Now I can check the summary of the model by using `model.summary()` . After creating the model I will import `ModelCheckpoint` and `EarlyStopping` method from **KERAS**. I will create an object of both `ModelCheckpoint` and `EarlyStopping` and pass that as callback functions to `fit_generator`. After that the model will start to train and I will start to see the training/validation accuracy and loss. Now I can visualize training/testing accuracy and loss using `matplotlib`. For ResNet50 the accuracy is 96.37%.

Figure 3.2.4 shows the architecture of ResNet.

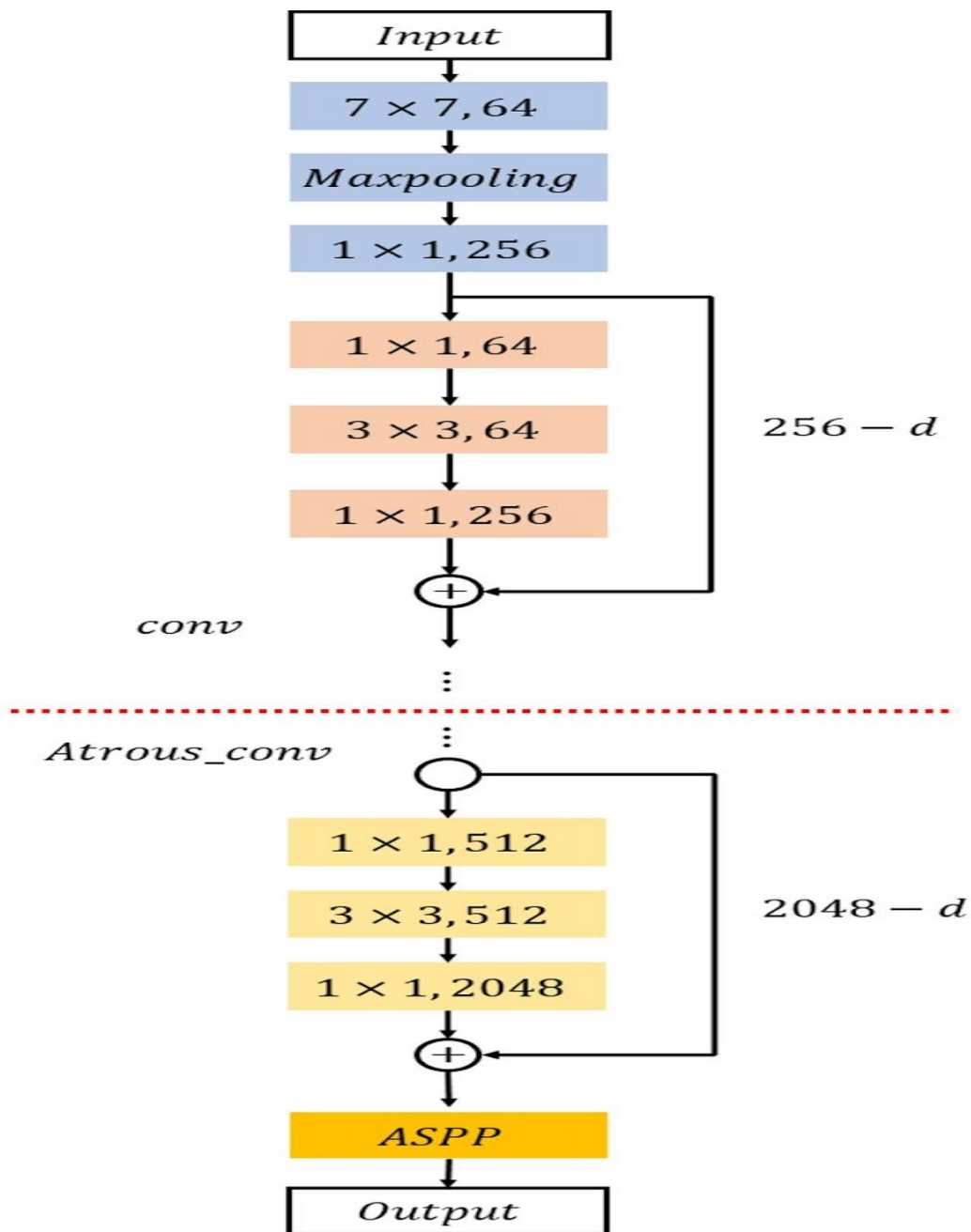


Figure 3.2.4 architecture of ResNet

- **Xception [12]:**

Xception is a convolution neural net (CNN) architecture. To implementing this architecture, at first I imported all the necessary libraries. Like **KERAS**, **OS**, **NUMPY** which I will need to implement ResNet50. I also imported **ImageDataGenerator** from

KERAS. Then I need preprocessing to import data with labels easily into model. Then I import Xception architecture from **KERAS** application. Then I create an object of **ImageDataGenerator** for training and testing data and passing the folder which has train data to the object trdata and test data to the object tsdata. The **ImageDataGenerator** will automatically label all the data inside each folder with its same padding. After initializing the model I add

- 2x convolution layer of 64 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 2x convolution layer of 128 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 256 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2

I also add Rectified Linear Unit (ReLU) activation to each layer. For which all the negative values are not passed to the next layer.

After creating the entire convolution I pass the data to the dense layer so for that we flatten the vector which comes out of the convolutions and add

- 1x dense layer of 4096 units
- 1x dense layer of 4096 units
- 1x dense Softmax layer of 10 units

After that my model is finally prepared. Now I need to compile the model.

I will use Adam optimizer to reach the global minima while training out model. Then I will specify the learning rate of the optimizer, here in this case it is set at 0.001. Now I can check the summary of the model by using model.summary() . After creating the

model I will import ModelCheckpoint and EarlyStopping method from **KERAS**. I will create an object of both ModelCheckpoint and EarlyStopping and pass that as callback functions to fit_generator. After that the model will start to train and I will start to see the training/validation accuracy and loss. Now I can visualize training/testing accuracy and loss using matplotlib. For Xception the accuracy is 92.41%.

- **InceptionV3 [13]:**

InceptionV3 is a convolution neural net (CNN) architecture. To implementing this architecture, at first I imported all the necessary libraries. Like **KERAS**, **OS**, **NUMPY** which I will need to implement ResNet50. I also imported **ImageDataGenerator** from **KERAS**. Then I need preprocessing to import data with labels easily into model. Then I import InceptionV3 architecture from **KERAS** application. Then I create an object of **ImageDataGenerator** for training and testing data and passing the folder which has train data to the object trdata and test data to the object tsdata. The **ImageDataGenerator** will automatically label all the data inside each folder with its same padding. After initializing the model I add

- 2x convolution layer of 64 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 2x convolution layer of 128 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 256 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2

I also add Rectified Linear Unit (ReLU) activation to each layer. For which all the negative values are not passed to the next layer.

After creating the entire convolution I pass the data to the dense layer so for that I flatten the vector which comes out of the convolutions and add

- 1x dense layer of 4096 units
- 1x dense layer of 4096 units
- 1x dense Softmax layer of 10 units

After that my model is finally prepared. Now I need to compile the model.

I will use Adam optimizer to reach the global minima while training out model. Then I will specify the learning rate of the optimizer, here in this case it is set at 0.001. Now I can check the summary of the model by using `model.summary()`. After creating the model I will import `ModelCheckpoint` and `EarlyStopping` method from **KERAS**. I will create an object of both `ModelCheckpoint` and `EarlyStopping` and pass that as callback functions to `fit_generator`. After that the model will start to train and I will start to see the training/validation accuracy and loss. Now I can visualize training/testing accuracy and loss using `matplotlib`. For InceptionV3 the accuracy is 92.41%. Figure 3.2.5 shows the architecture of Inception.

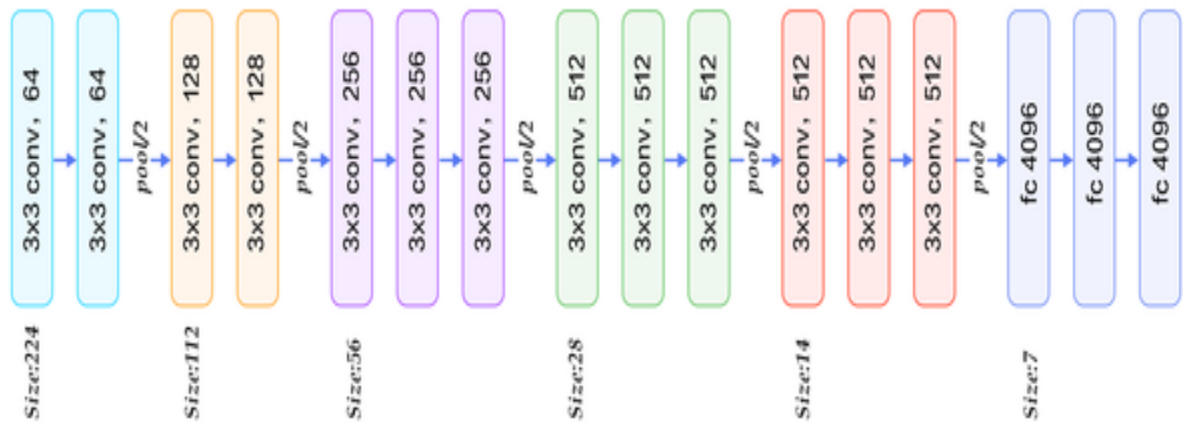


Figure 3.2.5 architecture of Inception

- **MobileNet [14]:**

MobileNet is a convolution neural net (CNN) architecture. To implementing this architecture, at first I imported all the necessary libraries. Like **KERAS**, **OS**, **NUMPY** which I will need to implement ResNet50. Here I will be using sequential method as I are creating a sequential model. I also imported **ImageDataGenerator** from **KERAS**. Then I need preprocessing to import data with labels easily into model. Then I import MobileNet architecture from **KERAS** application. Then I create an object of **ImageDataGenerator** for training and testing data and passing the folder which has train data to the object trdata and test data to the object tsdata. The **ImageDataGenerator** will automatically label all the data inside each folder with its same padding. After initializing the model I add

- 2x convolution layer of 32 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 2x convolution layer of 64 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 2x convolution layer of 128 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 256 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 2x convolution layer of 1024 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2

I also add Rectified Linear Unit (ReLU) activation to each layer. For which all the negative values are not passed to the next layer.

After creating the entire convolution I pass the data to the dense layer so for that I flatten the vector which comes out of the convolutions and add

- 1x dense layer of 1024 units
- 1x dense layer of 1024 units
- 1x dense Softmax layer of 10 units

After that my model is finally prepared. Now I need to compile the model.

I will use Adam optimizer to reach the global minima while training out model. Then I will specify the learning rate of the optimizer, here in this case it is set at 0.001. Now I can check the summary of the model by using `model.summary()`. After creating the model I will import `ModelCheckpoint` and `EarlyStopping` method from **KERAS**. I will create an object of both `ModelCheckpoint` and `EarlyStopping` and pass that as callback functions to `fit_generator`. After that the model will start to train and I will start to see the training/validation accuracy and loss. Now I can visualize training/testing accuracy and loss using `matplotlib`. For MobileNet the accuracy is 93.07%. Figure 3.2.6 shows the architecture of MobileNet.

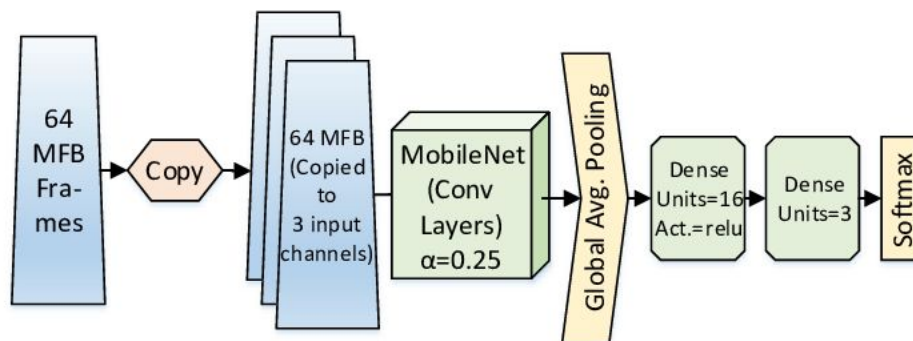


Figure 3.2.6 architecture of MobileNet

- **DenseNet121 [15]:**

DenseNet121 is a convolution neural net (CNN) architecture. To implementing this architecture, at first I imported all the necessary libraries. Like **KERAS**, **OS**, **NUMPY** which I will need to implement ResNet50. Here I will be using sequential method as I are creating a sequential model. I also imported **ImageDataGenerator** from **KERAS**. Then I need preprocessing to import data with labels easily into model. Then I import DenseNet121 architecture from **KERAS** application. Then I create an object of **ImageDataGenerator** for training and testing data and passing the folder which has train data to the object trdata and test data to the object tsdata. The **ImageDataGenerator** will automatically label all the data inside each folder with its same padding. After initializing the model I add

- 2x convolution layer of 64 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 2x convolution layer of 128 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 256 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 512 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2
- 3x convolution layer of 1024 channel of 3x3 **KERNEL** and use same padding
- 1x maxpool layer of 2x2 pool size and stride 2x2

I also add Rectified Linear Unit (ReLU) activation to each layer. For which all the negative values are not passed to the next layer.

After creating the entire convolution I pass the data to the dense layer so for that I flatten the vector which comes out of the convolutions and add

- 1x dense layer of 4096 units

- 1x dense layer of 4096 units
- 1x dense Softmax layer of 10 units

After that my model is finally prepared. Now I need to compile the model.

I will use Adam optimizer to reach to the global minima while training out model. Then I will specify the learning rate of the optimizer, here in this case it is set at 0.001. Now I can check the summary of the model by using `model.summary()`. After creating the model I will import `ModelCheckpoint` and `EarlyStopping` method from **KERAS**. I will create an object of both `ModelCheckpoint` and `EarlyStopping` and pass that as callback functions to `fit_generator`. After that the model will start to train and I will start to see the training/validation accuracy and loss. Now I can visualize training/testing accuracy and loss using `matplotlib`. For DenseNet121 the accuracy is 91.75%. Figure 3.2.7 shows the architecture of DenseNet.

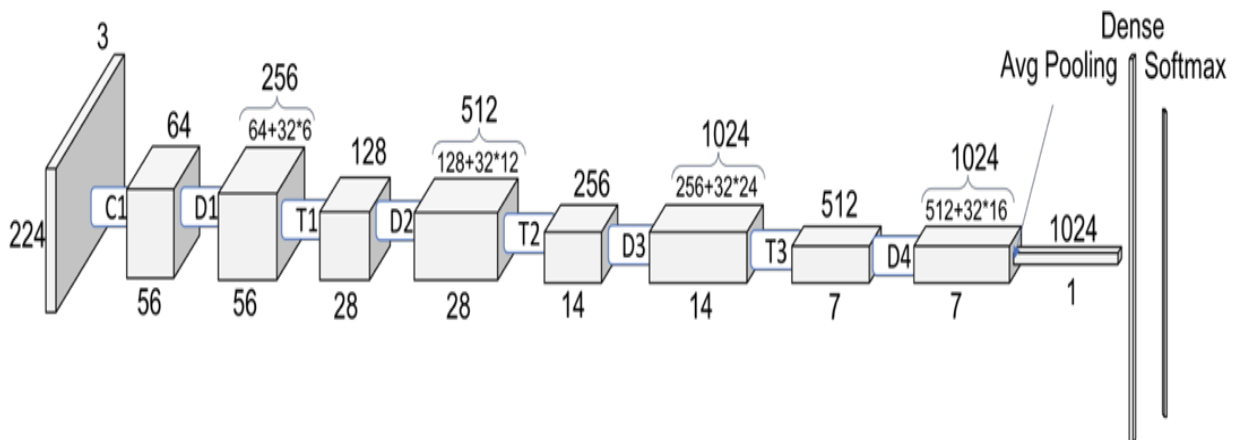


Figure 3.2.7 architecture of DenseNet

User Interface:

I simply develop software that will recognize Bangla Handwritten joint letter by implementing my selected algorithm.



Figure 3.2.8 Home page of the project

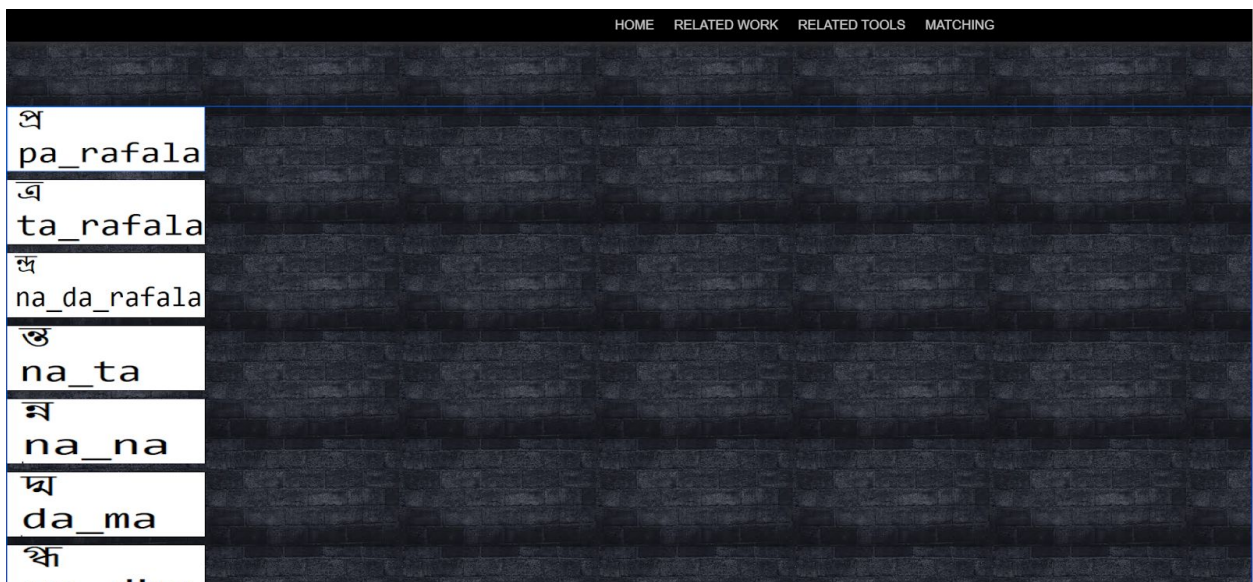


Figure 3.2.9 Joint Letter page

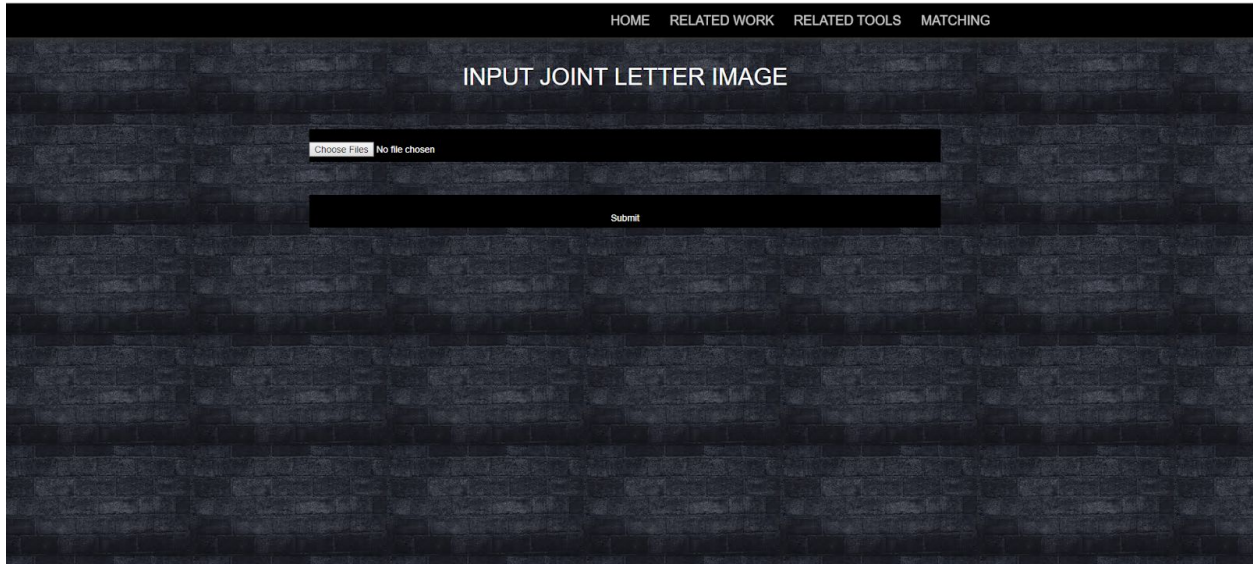


Figure 3.2.10 Matching(image input) page

Table	Action	Rows	Type	Collation	Size	Overhead
da_ma	Browse Structure Search Insert Empty Drop	160	InnoDB	latin1_swedish_ci	4.5 MiB	-
ga_dha	Browse Structure Search Insert Empty Drop	199	InnoDB	latin1_swedish_ci	5.5 MiB	-
img	Browse Structure Search Insert Empty Drop	214	InnoDB	latin1_swedish_ci	4.5 MiB	-
la_la	Browse Structure Search Insert Empty Drop	200	InnoDB	latin1_swedish_ci	5.5 MiB	-
matching	Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	32 KiB	-
ma_bha	Browse Structure Search Insert Empty Drop	200	InnoDB	latin1_swedish_ci	5.5 MiB	-
na_da_rafala	Browse Structure Search Insert Empty Drop	190	InnoDB	latin1_swedish_ci	5.5 MiB	-
na_na	Browse Structure Search Insert Empty Drop	3,259	InnoDB	latin1_swedish_ci	4.5 MiB	-
na_ta	Browse Structure Search Insert Empty Drop	2,162	InnoDB	latin1_swedish_ci	4.5 MiB	-
pha_rafala	Browse Structure Search Insert Empty Drop	190	InnoDB	latin1_swedish_ci	5.5 MiB	-
ta_rafala	Browse Structure Search Insert Empty Drop	199	InnoDB	latin1_swedish_ci	3.5 MiB	-
11 tables	Sum	6,979	InnoDB	utf8mb4_general_ci	49.2 MiB	0 B

Figure 3.2.11 Image of the project Database

3.3 Data Collection Procedure

For my target work, firstly I have collected my data. I have collected images of joint letters from multiple sources. The total dataset contains 172 types of joint letters. I gather total 183524 images of joint letters. In this project I have worked with 10 types of joint letters.

- Some images are collected from BornoNet dataset [16].
- Some pictures are collected from our Research Lab (UC).
- And some data are collected by the help our friends and family.

In this project initially we are working with only 10 types of joint letters. In our working dataset 200 images of Pa_rafala(প্র), 199 of Ta_rafala(ত), 2329 of Na_ta(ত), 190 of Na_da_rafala(দ), 3242 of Na_na(ন), 200 of Pha_rafala(ফ), 200 of M_bha(ম), 160 of Da_ma(ম), 179 images of Ga_dh(গ) are available.

3.4 Statistical Analysis

Table 3.4.1 the statistical analysis of each joint letter:

Table 3.4.1 statistical analysis

Joint Letter in Bangla	Class Name	Class Label	Total Data	Train Data	Test Data
প্র	pa_rafala	0	200	160	40
ত	ta_rafala	1	199	160	40
ত	na_ta	2	2329	1864	466
দ	na_da_rafala	3	190	152	39
ন	na_na	4	3242	2594	649
ফ	pha_rafala	5	200	160	41
ম	la_la	6	200	160	41
ম	ma_bha	7	200	160	41

ঝ	da_ma	8	160	128	33
ঞ	ga_dha	9	179	144	36

3.5 Implementation Requirements

I am researching about Handwritten Bangla Joint Letter recognition. List of tools, devices, languages and programs that we used in our project are given below:

- **Back end:**

1. Python (version 3.7.4)
2. CPU: Intel core i5
3. RAM: 4GB
4. Operating system: Windows 10 64 bit
5. MySQL database
6. Keras
7. PHP
8. OpenCV
9. Numpy
10. Tensorflow
11. Sklearn
12. Panda

Python is a programming language that supports functional and object-oriented programming. Numpy is a library for python. Numpy is used for matrix manipulation, calculation or some other scientific computing. Numpy is also used as multi-dimensional contains of generic data. openCV is another library for programming language that mainly used in real-time computer vision. OpenCV is a library that supports deep learning frameworks like tensorflow. In this project I use tensorflow in backend for deep learning . OpenCV also used for image processing. Sklearn is a tool in python that is comparatively simple and efficient for data analysis and data mining. In my project I use sklearn for splitting dataset into training and testing data. Keras is another python's

library that capable of running on top of tensorflow. It is an API which is designed for humans that provides clear and actionable feedback upon user error. Panda is a library function that provide flexible fast and expressive data. It is also used for data processing and for CSV file read write. For software development, I use PHP in backend because it is a server side scripting language for web development.

- **Front end:**

1. Bootstrap
2. HTML
3. CSS
4. JS

CSS is a stylesheet language that is used to describe the document which is written in HTML. it describes how elements should be rendered on screen or on the media. HTML is a markup language that is designed to be displayed in a web site. It can be assisted by javaScript or CSS. javaScript is a language that is object oriented for web development. It makes web development easier and more attractive. It mainly used to make a responsive web site. In my project I also use bootstrap which is a framework for making a website quickly and easily. It includes HTML and CSS file for forms, buttons, modals, navigations etc.

Finally we implemented this project in python to get the outcomes.

CHAPTER 4

EXPERIMENT RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, I will describe the experiment that I made. After reading this chapter anyone can get knowledge about the result of my project expected outcome and also get knowledge about the results of proposed model shows on recognition of Bangla joint letter based on each algorithm which is implemented in my project.

4.2 Experimental Results

- **Sequential Convolutional Neural Network:**

For sequential CNN model I get the accuracy rate 82.37%.

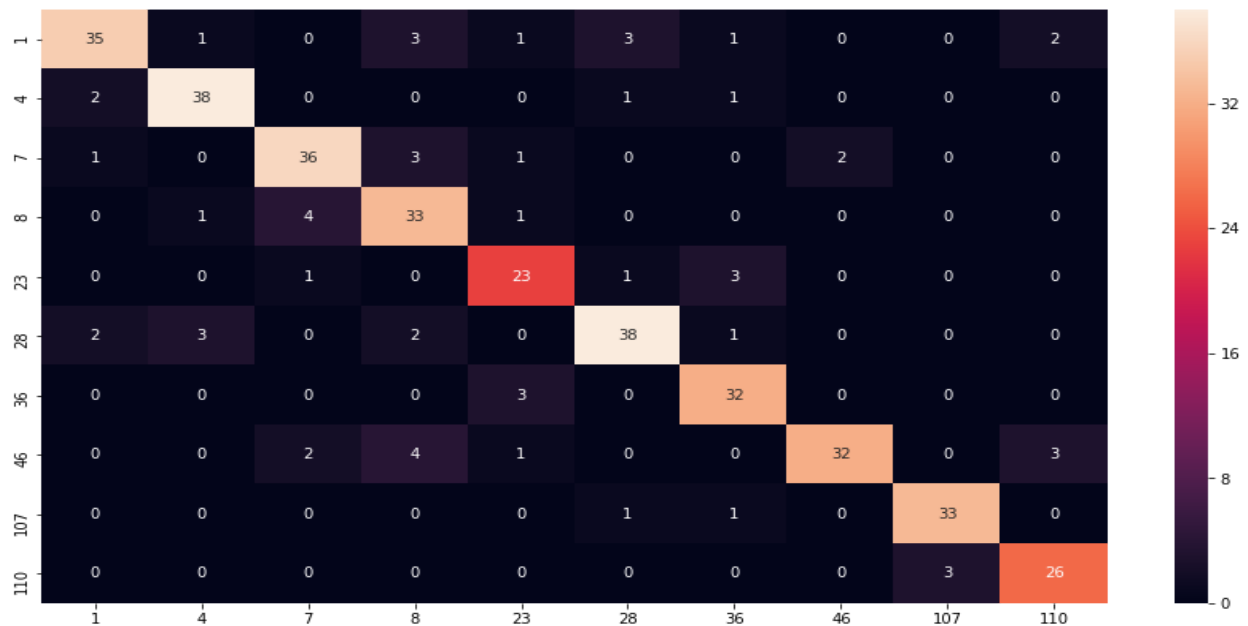


Figure 4.2.1 confusion matrix of CNN

- **Other Architectures of CNN :**

Table 4.2.1 Experimental Result of other architecture

Model	Size	Parameter s	Depth	Train Accuracy	Train Loss	Test Accuracy	Test Loss
Xception	88 MB	22,910,480	126	0.9638	0.1074	0.9241	0.3068
VGG16	528 MB	138,357,54 4	23	0.9673	0.1097	0.9604	0.0569
ResNet50	98 MB	25,636,712	-	0.9822	0.0587	0.9637	0.0221
InceptionV 3	92 MB	23,851,784	159	0.9660	0.1052	0.9241	0.3693
MobileNet	16 MB	4,253,864	88	0.9631	0.1194	0.9307	0.1019
DenseNet1 21	33 MB	8,062,504	121	0.9695	0.0791	0.9175	0.6726

VGG16

X-axis of the graph represent epoch value and y-axis represents the accuracy value.

Graph:

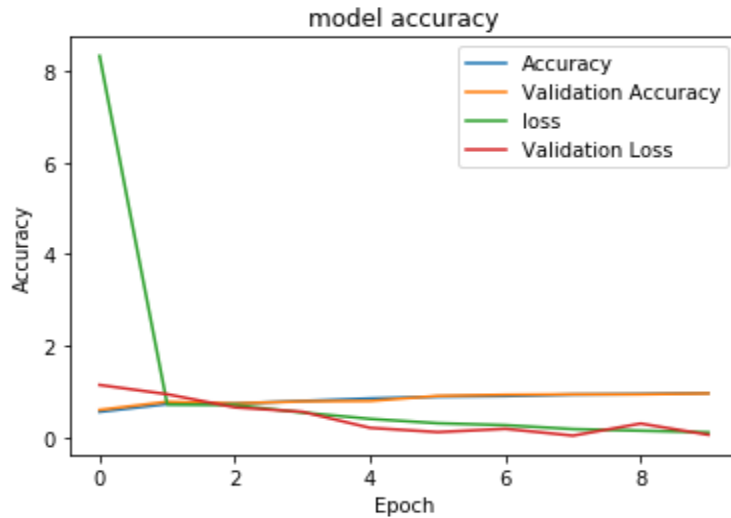


Figure 4.2.2 Graph of VGG16

Inception

Graph:

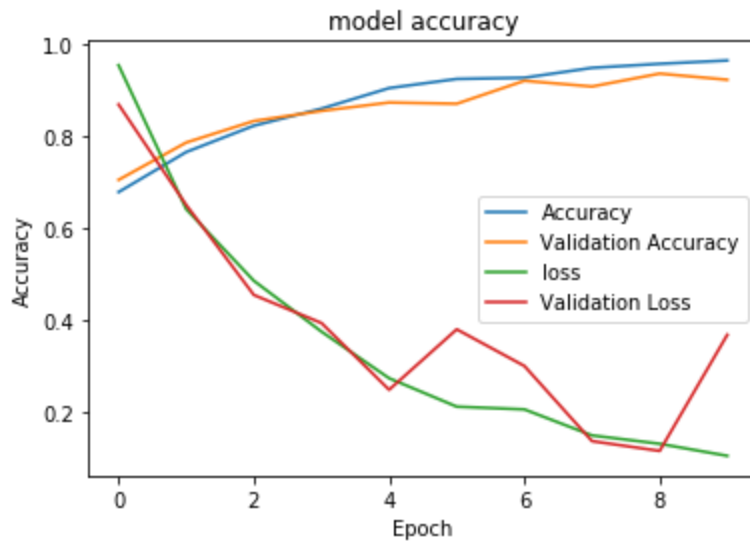


Figure 4.2.3 Graph of InceptionV3

Xception

Graph:

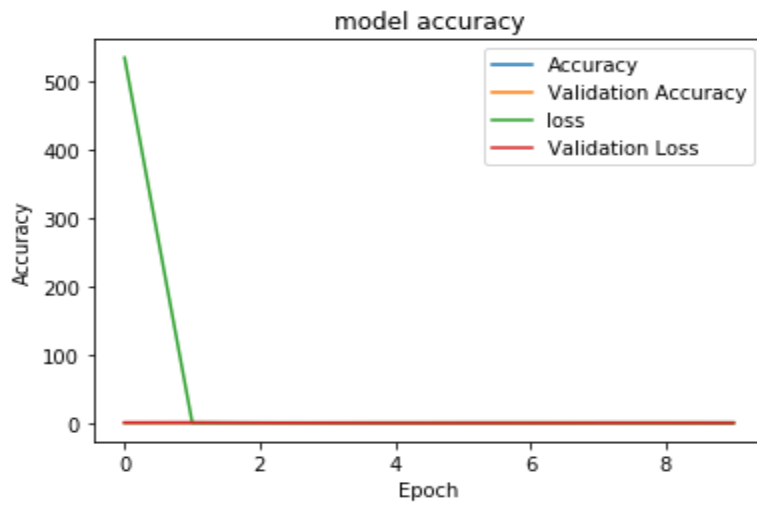


Figure 4.2.4 Graph of Xception

MobileNet

Graph:

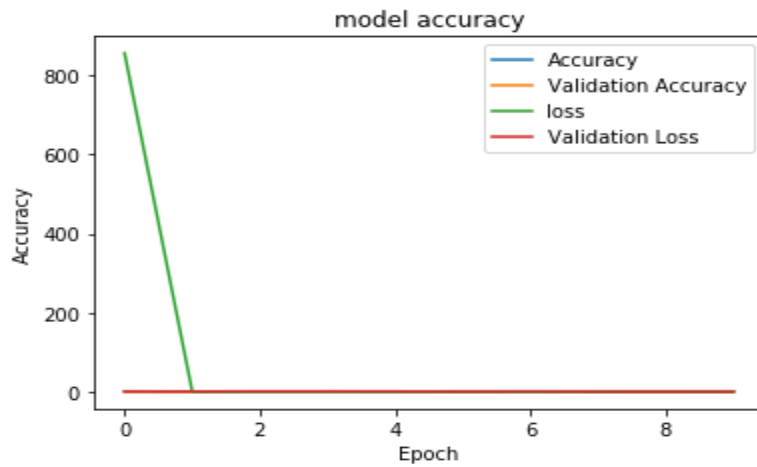


Figure 4.2.5 Graph of MobileNet

DenseNet121

Graph:

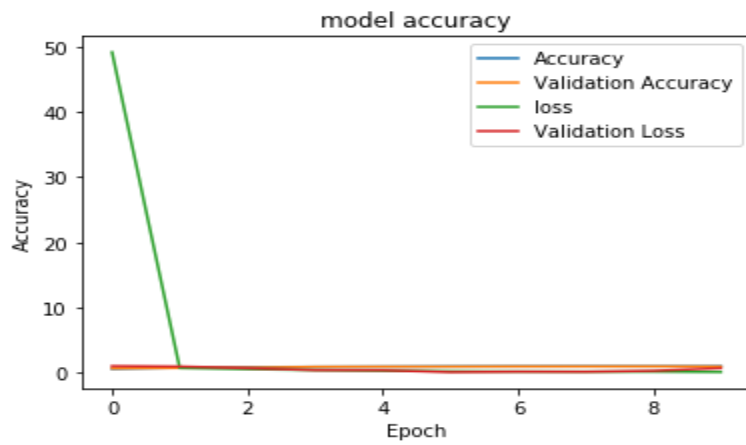


Figure 4.2.6 Graph of DenseNet121

ResNet50

Graph:

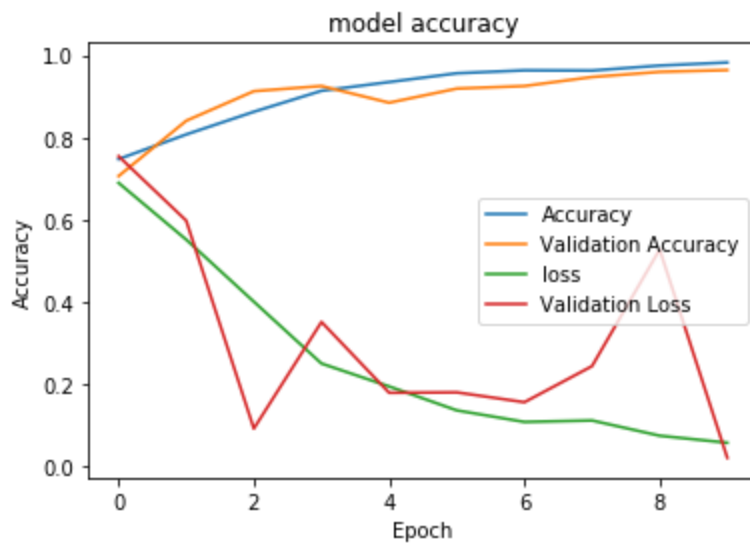


Figure 4.2.7 Graph of ResNet50

4.3 Descriptive Analysis

We used confusion matrix for accuracy visualization of CNN and for other architectures we plot graph. For Xception 92.41% accuracy, for VGG16 96.04% accuracy, for ResNet50 96.37% accuracy, for InceptionV3 92.41% accuracy, for MobileNet 93.07% accuracy, for DenseNet121 91.75% accuracy and for CNN the accuracy is 82.37%.

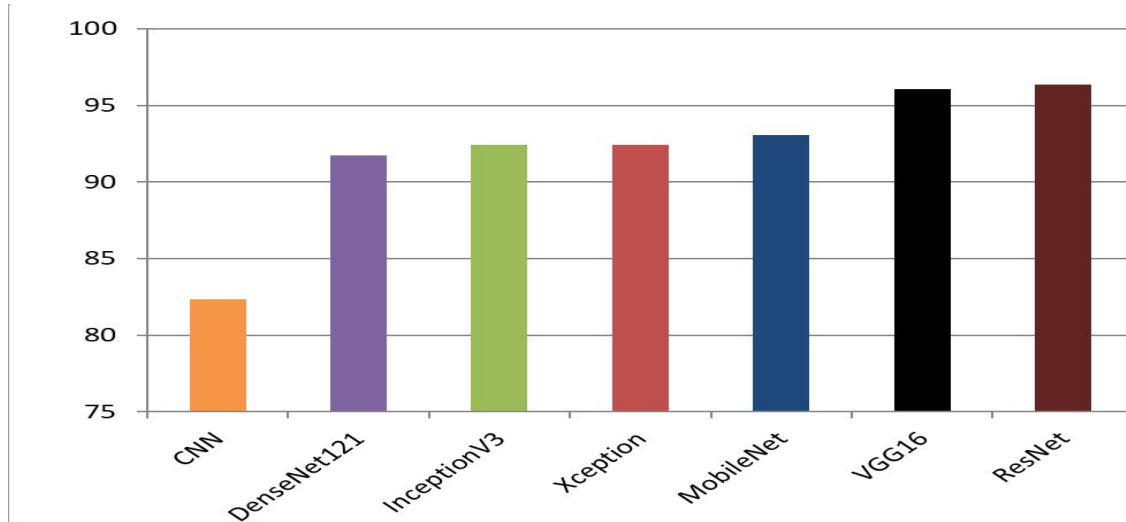


Figure 4.3.1 Testing accuracy for different architecture

For all architecture, we use the same steps per epoch as 100, validation step as 10 and epoch as 10 for all architecture. But the provided results are different in every model. It happened because every model has a different number of deep layers. For VGG16 network is 19 layers deep. VGG16 always uses 3 x 3 filters with stride of 1 in convolution layer and uses the same padding in pooling layers 2 x 2 with stride of 2. In ResNet50 the input matrix calculated in two linear transformations with ReLU activation function. ResNet directly copy the input matrix to the second transformation output and sum the output in final relu function. For ResNet network is 100 layers deep. For Inception network is 22 layers deep. For MobileNet network is 54 layers deep. In DenseNet each block contains a set of conv blocks depending on the architecture depth. Some other works are done are also done on bangla joint letter recognition. A table 4.3.1 is given below that shows the comparative performance

Table 4.3.1 comparison with other works

Reference	Technique	Character	Accuracy
This work	Transfer learning using ResNet50	Bangla joint letter	96.37%
Asfi Fardous, Shyla Afroge2019[17]	CNN	Bangla joint letter	95.5%

4.5 Summary

After getting the result, we can see 92.41% accuracy for Xception, 96.04% accuracy for VGG16, 96.37% accuracy for ResNet50, 92.41% accuracy for InceptionV3, 93.07% accuracy for MobileNet, 91.75% accuracy for DenseNet121 and for sequential CNN accuracy is 82.37%. So our proposed model is ResNet50 as the accuracy of it's better than other models.

CHAPTER 5

Summary, Conclusion, Recommendation and Limitation

5.1 Summary of the Study

Bangla is the mother language of Bangladesh, apart from it is the official language of Bangladesh and Bangla is the seventh most spoken language in the world. As it is such a popular language, there is a wide area for research. In our project, I work with Bangla joint letter recognition. Here I implemented some deep convolutional neural networks (DCNNs) for handwritten Bangla joint letter. Then we compared the results and proposed a model that provides the best result. And that is ResNet50 architecture. In this project, I have used a dataset of 172 different category joint letters. Here I work with only 10 different category joint letters consist of 7099 images.

5.2 Conclusions

In this project, we compared the performance of several popular deep convolutional neural networks (DCNNs) for handwritten Bangla joint letter. And we get 92.41% accuracy for Xception, 96.04% accuracy for VGG16, 96.37% accuracy for ResNet50, 92.41% accuracy for InceptionV3, 93.07% accuracy for MobileNet, 91.75% accuracy for DenseNet121 and for sequential CNN the accuracy is 82.37%. The experimental results indicated that ResNet50 is the best for recognizing handwritten Bangla joint letter. In future, some other DCNN models, such as NasNetMobile, NiN, FractalNet etc. will be implemented to improve the result of handwritten Bangla joint letter recognition.

5.3 Recommendations

There are so many deep convolutional neural networks. For now, we just implemented CNN and other 6 popular architectures. We get 82.37% accuracy to 96.37% accuracy. The joint letter recognition is achieved through training and testing models.

- Result can be improved by increasing epoch size
- Using a more efficient algorithm can also improve the result

5.4 Limitation

This time I work with 10 joint letter characters only. And I implemented MobileNet, InceptionV3, Xception, DenseNet, VGG16, ResNet50 architecture and compared those results. In future will work with some other architecture, like NasNetMobile, NiN, FractalNet etc. And that time we will work with our full dataset that has 172 joint letter characters. We will also try to develop a model that will recognize Bangla joint letter more perfectly. This project can recognize joint letter from image of single joint letter. In future I will explore the model that will be capable

of recognizing from paragraph and then convert the paragraph from bangla to english.

Appendices

Appendix A: Research Reflection

Having finished this project, I get a higher knowledge on this topic than previous. After analyzing the experimental results of this project and exploring related works, I have learned that it is possible to recognizing Bangla handwritten joint letter accurately by using Deep Neural Network (DNN) with a developed dataset. If I had another chance to work with Bangla joint letter, than that time I will implement some other popular Deep Neural Networks (DNN) for better results.

Appendix B: Related Issues

One of the important issues is collecting data and preparing dataset. There are some difficulties that I faced while implementing this project. Firstly for fixing image dimension for resizing function. And when implementing algorithms and architectures, some of the architecture took so long time for running model. That makes my system slower. A common problem is building an appropriate model that can recognize enormous variety of handwriting.

Reference:

- [1] Bangla language information available at: https://en.wikipedia.org/wiki/Bengali_language [last accessed on 20-09-2019]
- [2] Bangla joint letter available at: <https://forum.daffodilvarsity.edu.bd/index.php?topic=11714.0> [last accessed on 24-09-2019]
- [3] Introducing the Boise State Bangla Handwriting Dataset and an Efficient Offline Recognizer of Isolated Bangla Characters paper. Available at: Majid, N., & Smith, E.H. (2018). Introducing the Boise State Bangla Handwriting Dataset and an Efficient Offline Recognizer of Isolated Bangla Characters. *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 380-385. [last accessed on 25-09-2019]
- [4] A Novel Approach of Bangla Handwritten Text Recognition using HMM paper. Available at: Roy, P.P., Dey, P., Roy, S., Pal, U., & Kimura, F. (2014). A Novel Approach of Bangla Handwritten Text Recognition Using HMM. *2014 14th International Conference on Frontiers in Handwriting Recognition*, 661-666. [last accessed on 27-09-2019]
- [5] Bangla/English Script Identification Based on Analysis of Connected Component Profiles paper. Available at: Zhou, L., Lu, Y., & Tan, C.L. (2006). Bangla/English Script Identification Based on Analysis of Connected Component Profiles. *Document Analysis Systems*. [last accessed on 30-09-2019]
- [6] BornoNet: Bangla Handwritten Characters Recognition Using Convolutional Neural Network paper. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050918321240> [last accessed on 2-10-2019]
- [7] Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks paper. Available at: Alom, M.Z., Sidike, P., Hasan, M., Taha, T.M., & Asari, V.K. (2017). Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks. *Comp. Int. and Neurosc.* [last accessed on 6-10-2019]
- [8] Bangla Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation paper. Available at: Chowdhury, R., Hossain, M.R., Islam, R.U., Andersson, K., & Hossain, S. (2019). Bangla Handwritten Character Recognition using Convolutional Neural Network with Data Augmentation. *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, 318-323. [last accessed on 9-10-2019]
- [9] CNN available at: <https://stackoverflow.com/questions/54943307/create-cnn-model-architecture-diagram-in-keras> [last accessed on 12-10-2019]
- [10] VGG16 available at: <https://neurohive.io/en/popular-networks/vgg16/> [last accessed on 19-10-2019]

- [11] ResNet50 available at:
https://www.researchgate.net/figure/ResNet-architecture-with-atrous-convolution_fig2_328099227 [last accessed on 21-10-2019]
- [12] Xception available at:
<https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568> [last accessed on 21-10-2019]
- [13] InceptionV3 available at:
<https://medium.com/coinmonks/rethinking-the-inception-architecture-for-computer-vision-part-1-2938cc7c7872>
[last accessed on 21-10-2019]
- [14] MobileNet available at:
https://www.researchgate.net/figure/Network-Architecture-of-MobileNet_fig1_328654938 [last accessed on 21-10-2019]
- [15] DenseNet121 available at:
<https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a> [last accessed on 21-10-2019]
- [16] BornoNet database available at: <https://shahariarrabby.github.io/ekush/#external> [last accessed on 22-10-2019]
- [17] Handwritten Isolated Bangla Compound Character Recognition paper. Available at:
<https://ieeexplore.ieee.org/document/8679258> [last accessed on 23-10-2019]

161-15-681

ORIGINALITY REPORT

12%

SIMILARITY INDEX

3%

INTERNET SOURCES

7%

PUBLICATIONS

6%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	2%
2	www.comp.nus.edu.sg Internet Source	1%
3	Md Zahangir Alom, Paheding Sidike, Mahmudul Hasan, Tarek M. Taha, Vijayan K. Asari. "Handwritten Bangla Character Recognition Using the State-of-the-Art Deep Convolutional Neural Networks", Computational Intelligence and Neuroscience, 2018 Publication	1%
4	Akm Shahariar Azad Rabby, Sadeka Haque, Sanzidul Islam, Sheikh Abujar, Syed Akhter Hossain. "BornoNet: Bangla Handwritten Characters Recognition Using Convolutional Neural Network", Procedia Computer Science, 2018 Publication	1%
5	mc.ai Internet Source	1%

6	<p>Nishatul Majid, Elisa H. Barney Smith. "Introducing the Boise State Bangla Handwriting Dataset and an Efficient Offline Recognizer of Isolated Bangla Characters", 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2018 Publication</p>	1%
7	<p>Submitted to King's College Student Paper</p>	1%
8	<p>Roy, Partha Pratim, Prasenjit Dey, Sangheeta Roy, Umapada Pal, and Fumitaka Kimura. "A Novel Approach of Bangla Handwritten Text Recognition Using HMM", 2014 14th International Conference on Frontiers in Handwriting Recognition, 2014. Publication</p>	<1%
9	<p>dspace.daffodilvarsity.edu.bd:8080 Internet Source</p>	<1%
10	<p>Submitted to Embry-Riddle Aeronautical University Student Paper</p>	<1%
11	<p>Rismiyati, Khadijah, Adi Nurhadiyatna. "Deep learning for handwritten Javanese character recognition", 2017 1st International Conference on Informatics and Computational Sciences (ICICoS), 2017 Publication</p>	<1%

12	Submitted to Kingston University Student Paper	<1%
13	"Recent Trends in Image Processing and Pattern Recognition", Springer Science and Business Media LLC, 2019 Publication	<1%
14	Submitted to Sri Lanka Institute of Information Technology Student Paper	<1%
15	Submitted to University of Brighton Student Paper	<1%
16	collections.mun.ca Internet Source	<1%
17	ltu.diva-portal.org Internet Source	<1%
18	link.springer.com Internet Source	<1%
19	Submitted to Nanyang Polytechnic Student Paper	<1%
20	Submitted to Higher Education Commission Pakistan Student Paper	<1%

Exclude quotes

On Exclude bibliography

On