Bangali Handwritten Characters Classification Using Deep Convolutional Neural Network

by

Shihab Uddin Sikder
18301093
Md. Shafiul Muslebeen
18301116

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Shihab	Shafiul
Shihab Uddin Sikder 18301093	Md. Shafiul Muslebeen 18301116

Approval

The thesis titled "Bangali Handwritten Characters Classification Using Deep Convolutional Neural Network" submitted by

1. Shihab Uddin Sikder (18301093)
2. Md. Shafiul Muslebeen (18301116)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on May 19, 2022.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-supervisor:
(Member)

Ramkrishna Saha

Ramkrishna Saha
Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Handwritten letter classification of any given language has the potential to be used in various fields such as literature, educational institutions, digitization of government records etc. Bengali language with its complex sets of mixed characters, poses significant complexities in terms of automatic recognition of characters. In the Bengali character set, there are over 360 distinct characters among which a lot of similarities are present between different characters. Thus, the classification of these characters gets harder as the recognition system incorporates all these distinct characters. In recent years, a lot of research has been done to solve this problem on isolated datasets with significant results. Continuing the advancement in image processing, In this paper, we have proposed a custom CNN model which has been trained on Bangla Lekha Isolated dataset containing 1,66,106 images belong to 84 distinct classes with the capability to detect individual handwritten Bengali letters including digits, vowels, consonants and compound characters with 93.15% accuracy while using less number of parameters compared to existing popular models

Keywords: Deep learning, CNN, Bangali Characters, Image processing, DCNN, Handwritten Character Recognition, Bengali letters, Bengali compound characters.

Acknowledgement

We have extensively used the BanglaLekha Isolated Dataset as our primary dataset in this paper. We would like to acknowledge the authors of BanglaLekha Isolated Dataset. The work that the authors have put forth has allowed us to conduct this research. We would also like to express our deepest gratitude towards our supervisor Dewan Ziaul Karim and co-supervisor Ramkrishna Saha for their immense help in guiding us throughout the research. Without their help it would not have been possible for us to finish this paper. Also we would like to express our sincere regards to Md. Golam Rabiul Alam Sir for allocating us access to high performance computers in the research lab in which we have run our machine learning models. Lastly, we would like to express our sincere gratitude towards almighty Allah who has kept us alive and healthy during this pandemic and enabled us to successfully conduct this research.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN	Convolutional Neural Network
DT	Decision Tree
GNB	Gaussian Naive Bayes
GPU	Graphics processing unit
KNN	K-nearest Neighbour
LR	Logistic Regression
ReLU	Rectified Linear Activation Function
ResNet	Residual Neural Network
RF	Random Forest
RGB	Red Green Blue
SVM	Support Machine Vector

Chapter 1

Introduction

Handwriting is one of the most ancient and effective mediums in the process of humans sharing knowledge among each other. Handwriting was invented out of the necessity of keeping records of acquired knowledge. Historically, the tools used to write things down were not convenient. Initially, Stones, leaves, woods were used as the materials to hold the writings which was inconsistent and posed a lot of problems in long term maintainability of those records. As the need to share information grew dramatically, researchers and scientists devised new methods for storing information efficiently. During the Han Dynasty circa (25-220CE) paper was first invented which accelerated the whole process of written communication. Since then, written documents on paper have become the essential form of communication. In the modern era, computers have become the quintessential tool accompanying humans in every strata of life. Advances in computer image processing have enabled computers to process visual information that was unimaginable in the past. Nowadays, computers can be used to analyze various types of image data. One of the applications of image processing is to analyze human writings on paper and convert them into digital data that can be processed by computers. This conversion process is known as letter or text classification. The differences between handwritten characters and the ones represented in modern computers is that, handwritten characters vary in both shape and size based on individual writing styles while modern computers have a specific set of predefined characters that is consistent. The translation between what is human readable to machine readable is the main goal of text classification which can be achieved by using different machine learning techniques. Deep convolutional neural networks is a subset of machine learning that is proven to be effective in text classification. The Bangla script is the second most widely used script in the Indian subcontinent. Bangla is spoken as a primary or second language by around 200 million people. Therefore, the relevance of automated identification of Bangla written letters is enormous. In this research, we aim to use deep convolutional neural networks to classify Bengali handwritten letters.

1.1 Aim and objective

Through our research, we aim to build a system that can automatically detect handwritten letters in the Bengali language by using cutting edge machine learning algorithms.
To compile shortly, our research will focus on the following key areas:

- Analyze different convolutional neural networks to determine the most suitable algorithm in text detection.
- Apply the algorithm in data sets collected from the real world
- Detect the reliability of given algorithms by cross checking the outputs of the algorithms with correct outputs.
- Based on the test results, pinpoint the areas of improvement.
- Incorporate the new findings in to the existing algorithms and publish the new findings.

Chapter 2

Related Work

Lots of research has been done in regards to Bengali letter classification. The first ever research was done by Roy *et al* [1]. Lots of researcher has followed his path later on. Most of them improve the accuracy by applying different method like [2]–[9]. Academicians widely use CNN in this regard for its excellent performance in character classification compared to other classifiers. Convolutional Neural Networks reduce classification complexity by using a multilayer perceptron model that includes an input layer. Two or more hidden convolutional layers, pooling and normalization layers and the last layer is an output layer. Every layer works on a specific portion of the image. Then it forwards the result in the next layer if the portion of the image passes through its classifier. It has a very high potential in digit reorganization tasks. It was used on the MNIST dataset and gave a very good accuracy. M M Rahman *et al.* [10] classified the Bangla basic characters using (CNN). They used it on the CMATER database and successfully classified the Bangla basic characters with an accuracy of 85.96 percent. Chatterjee S. *et al.* [11] used a specific model, namely the Resnet50 model, pre-trained with the ILSVRC dataset and later used Transfer Learning to use the existing model in Bengali characters. Alif M. *et al.* [12]also took a step that is very much alike but instead with the Resnet18 model and Adam and Rmsprop optimizer. Begum H. *et al.* [13] used three main feature extraction techniques, namely: (i) Longest Run features, (ii) Histogram feature from chain code, and (iii) Gabor wavelet-based features. The authors then applied different common classifier algorithms on these extracted features to determine the best feature extraction technique. They achieved 76.47 percent accuracy on a Longest Run, and Chain Code Histogram methods are combined.

Shyla Afroge *et al.* suggested an example for Optical Character Recognition in Bengali based on Discrete Fréchet Distance. The model works very fine as it ends up giving 90-95 percent accuracy for both digit and characters [14]. According to a study, the feature based on a geometric convex hull can successfully detect solitary handwritten Bangla. The feature can detect both digits and characters with an accuracy of 99.45 percent [15].

KNN (K Nearest Neighbor), also known as the simplest classification algorithm, is also a very popular method in BHCR. It gives results based on the difference among the given data. According to certain studies, conventional or classic KNN offers good accuracy 95 percent for BHCR [16]. But it performs very poorly and ends up giving accuracy of 68.3 percent for a given dataset by Md Nazmul Hoq *et al.* [17]. In BHCR lots of features were used such as Gaussian filters, CNN combination dropout, and Gabor filters. With 98.78 percent accuracy, Gabor filters and CNN combination dropout features outperforms SOTA (state of the art) methods for HDBR [18].

Using Convolutional neural networks (ConvNet), Shopon et al. achieve the highest accuracy so far which is 99.50 percent [19]. For this process, they took 19,313 pictures from the ISI handwritten character dataset and trained ConvNet on it. Later, they tested it on the CMATERDB dataset. Although all of the applied approaches in the literature obtained high accuracy, they omitted to provide comparative research and a clear picture of the performance of distinct categorization algorithms. One of their previous studies looked at the performance of classification algorithms using Bangla Numeric Digit images [17].

DCNN was used in these [20], [21] articles for BHCR, with the maximum accuracy of 97.21 percent on 84 classes and 97.73 percent on 122 classes, respectively. DCNN dominated handwritten character recognition in BHCR and other languages. One of the most prominent datasets is the MNIST dataset. By regularizing neural networks using drop connect, this method can recognize English handwritten digits with a 99.79 percent accuracy [22]. In Bangla digit recognition, an autoencoder made from a neural network with DCNN achieved a high accuracy of 99.50 percent [23].

Bishwajit Purkaystha *et al.* [24] suggest recognizing Bangla Handwritten characters using the deep convolutional model. They tested the model or system on the BanglaLekha-Isolated dataset and got a very impressive accuracy. With an accuracy of 98.66 percent, it works very well on the numerical alphabet. Besides this, it gives 94.99 percent on vowel characters and 91.60 percent on compound characters. Lastly, it gives a 91.23 percent accuracy on alphabets. On average, it produces 89.93 percent accuracy on any Bangla characters.

Chapter 3

Dataset

3.1 Data Collection

Data collection is one of the most challenging works while doing research. A model can be particularly efficient if it has a big amount of data. Because of the lockdown and shutdown of educational institutions, we were under added strain. We couldn't get any data from people directly. Fortunately, some of the datasets were already freely available on several websites. Bangla Lekha-Isolated [25] is the most widely utilized and biggest dataset currently accessible. The following Table shows the number of images collected from each dataset. We divided the data into four groups for easier understanding: modifiers, fundamental characters, numerical digits, and certain often used compound characters images. In Bangla Lekha-Isolated we have found 98,950 basic characters, 47407 compound characters and 19748 numeral images.

Dataset	Basic Characters	Digits	Compound	Total
Bangla Lekha Isolated	98950	47407	19748	166105

Table 3.1: Class breakdown of Bangla Lekha Isolated Dataset

In Bangla Lekha Isolated dataset, it includes 50 Bangla basic characters, 10 Bangla numerals, and 24 chosen compound characters as sample.

3.2 Data Labeling and Naming Format

Each sample of the Bangla Lekha Isolated dataset contains a unique form ID that may be used to identify the participants' age, gender, district, and institute. Therefore a 22-digit long ID was used to label them. Among the 22 digits, the first 2 digits represent the district, next 4 digits represent the institution, next 1 digit represent gender, next 2 digit represent age, next 4 digits represent the date of the collection and last 4 digits represent the serial number of the form which was used to collect the data from people. And for better understanding each and every information were separated by an underscore . For example,

01-0002-1-18-1218-0928

Here the number 01 denotes that the participant is from Comilla, whereas 0002 denotes that the person is from Comilla zilla school and the next 1 means the participant is female and 18 denotes the participant's age is 18. 1218 denotes the data was collected in December 2018 and lastly 0928 denotes it was collected from 0928 number form.

3.3 Annotation

The data needed to be annotated since we intended to run supervised machine learning algorithms on it. So, after labeling the data with a 22 long digit an extra 1 or 2 digits was also added. As we have mentioned before the dataset has a total 84 classes (50 basic characters ,10 numbers,24 compound characters). The first 11 class was the Vowels which are also known as স্বরবর্ণ in Bangla. The data was annotated for them from numbers 1 to 11 increasingly. The next 39 class was Consonants which are also known as বযঞ্জনবর্ণ in Bangla. The data was annotated for them from numbers 12 to 50 increasingly. Like every other language, Bangla language has its number system which are also known as সংখ্যা in Bangla. The data was annotated for them from numbers 50 to 60 increasingly. A special case which makes Bangla letters different from most other languages is the compound characters. When two or more consonant make a one and unique character we call it as compound character which is also known as যুক্তবর্ণ in Bangla. The process of annotation is considered as one of the most important and difficult tasks. A simple annotation mistake can make the algorithm give the wrong result. The annotation has a very good Fleiss Kappa Interrater Agreement score.

3.4 Exploratory Data Analysis

Bangla is considered as the 4th largest spoken language in the world. Beside it's spoken usage it is largely used in writing too. Bangla has 11 vowels and 39 consonants and beside this a special class is also there in Bangla Alphabet which are called compound characters. Compound characters are a mix of two or more consonant characters. But the writing structures of compound letters are different and which make each and every character of Bangla alphabet as unique as possible. Some of the general characteristics of each major class, statistical and visual representation of the collected data (from Bangla Lekha Isolated) are given below.

3.5 Visual Representation

The following figures represent the class breakdown of Bangla Lekha Isolated dataset.

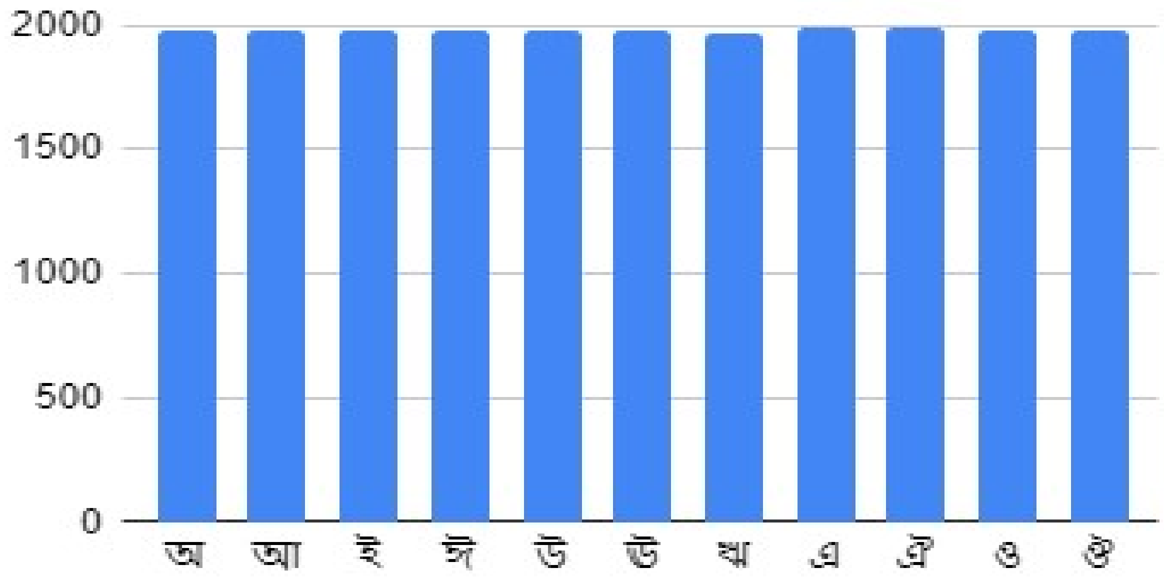


Figure 3.5.1: Frequency of Vowels in the Dataset

In the first figure 3.5.1, the distribution of each vowel character is shown and we can see the data are almost equally distributed. And the total number of vowel data is 21670.

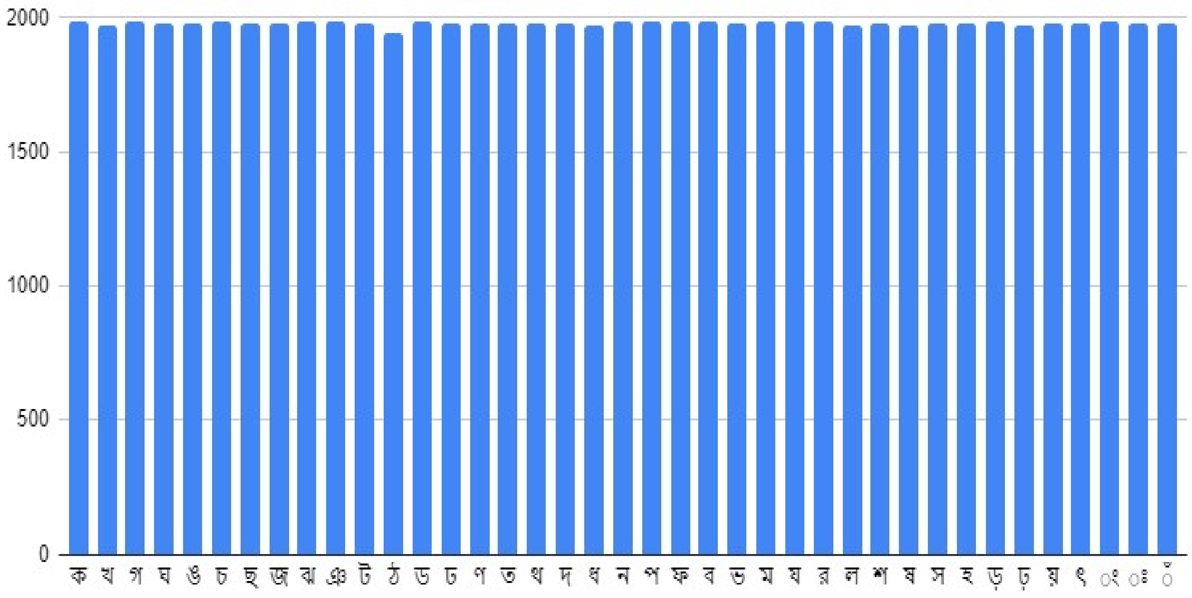


Figure 3.5.2: Frequency of Consonant in the Dataset

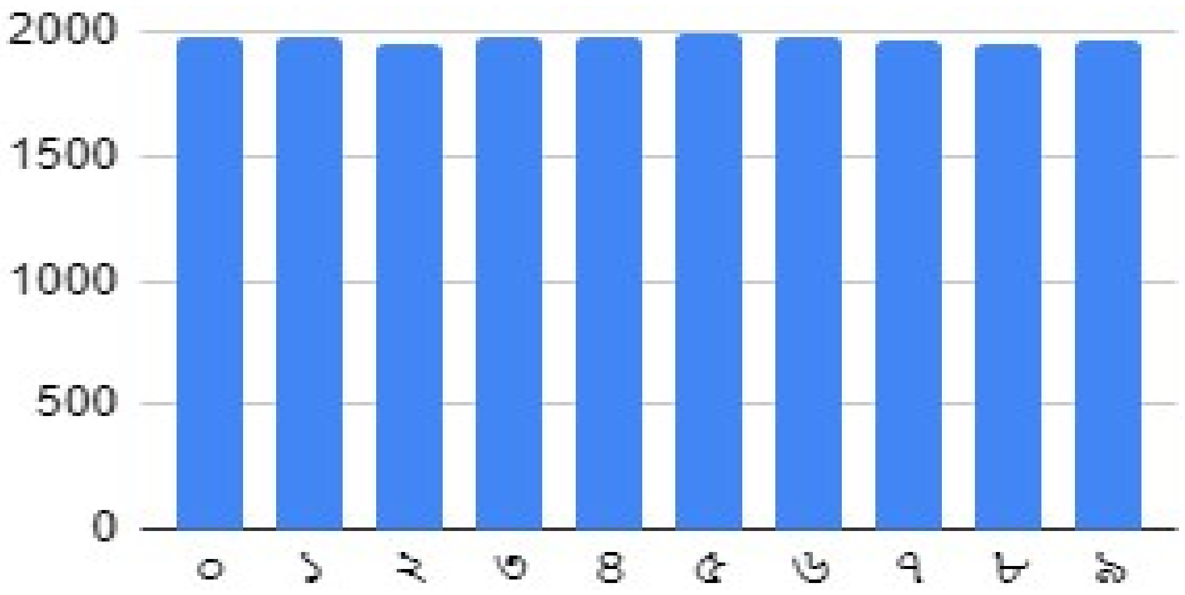


Figure 3.5.3: Frequency of Digits in the Dataset

Same distribution goes for the consonant and digit class and both are having equally distributed data shown in figure 3.5.2 and 3.5.3. The current pre-processed data for these two classes is 76,760 and 19,888 respectively.

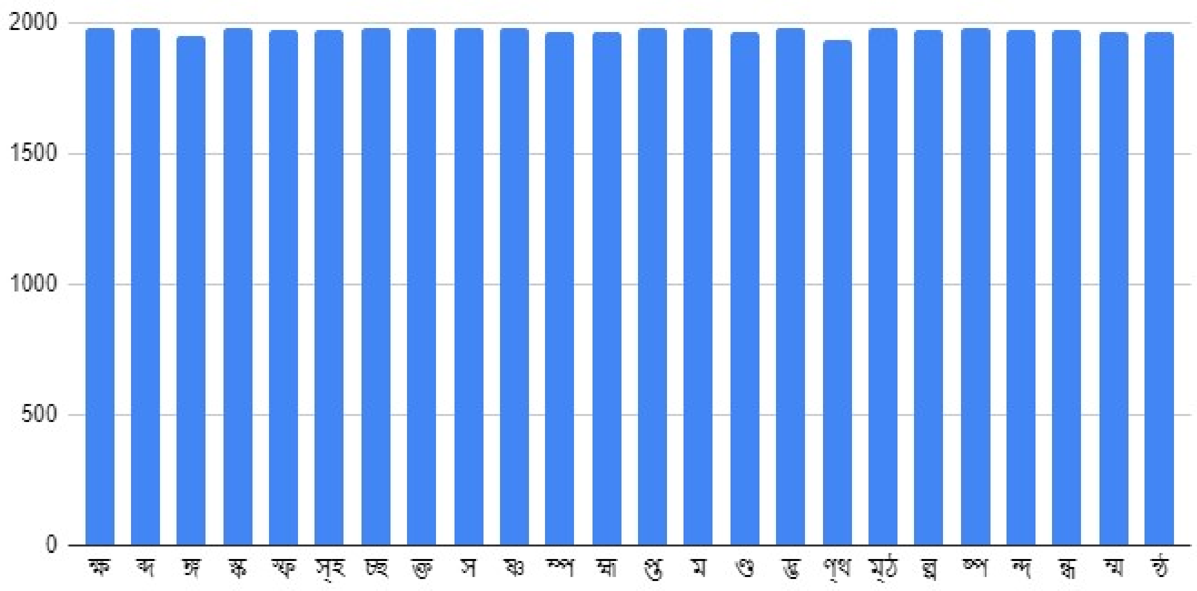


Figure 3.5.4: Frequency of Compounds in the Dataset

But the compound class's data are not equally distributed at all compared to other classes. People are most likely to make mistakes while writing the compound characters because of its complex structure. To remove this distribution problem, the BLI team deleted the wrong characters and labeled them accordingly. Figure 3.5.4 shows the distribution of each selected compound charters frequency.

General Statistic: The data also collected from people of different ages. As we know, people of different ages have different handwriting styles. 20 years ago, people almost did any official work by handwritten paper rather than a computer or typing machine. And people nowadays are almost fully dependent on computers while it comes to writing any official work. So, there is a significant style difference between the 25- and 7-year-olds. And beside this the student class mostly participates in writing by hand most of the time. So, the main goal was to take as much student handwriting as possible. The following figure shows the age class breakdown. And the male- female ratio was 60:40 percentage because both male and female participants had different handwriting styles. Figure 3.5.5 shows the age and male: female ratio breakdown.

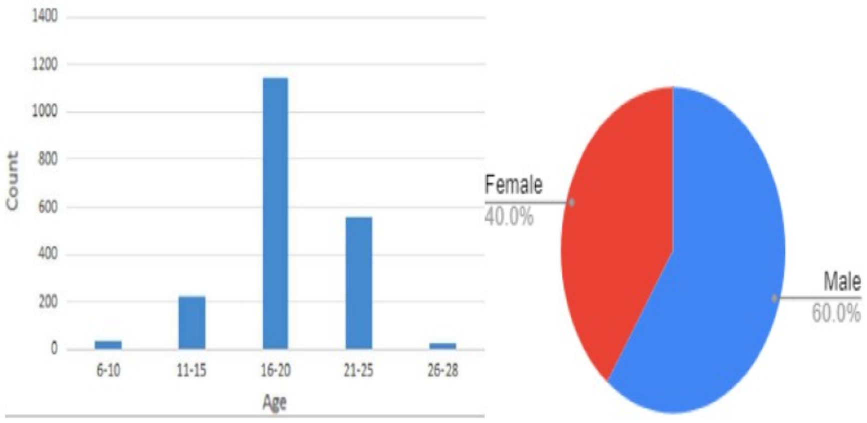


Figure 3.5.5: Frequency of different age of participants and male female ratio

Chapter 4

Data Pre-Processing

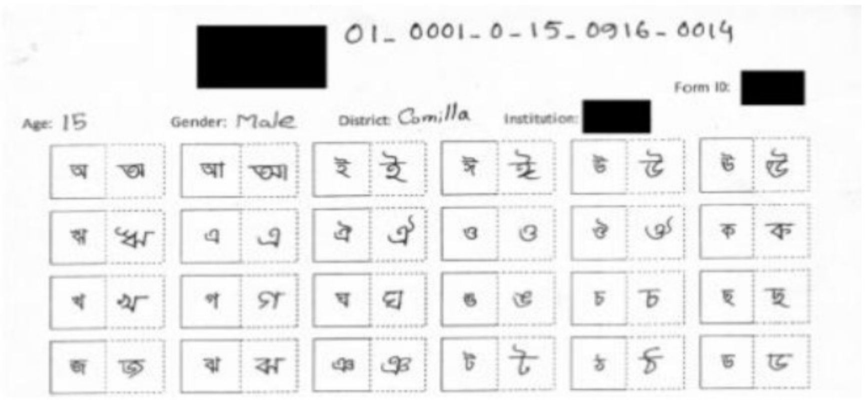
For our proposed method data pre-processing is the first step in character recognition and is critical in determining the recognition rate. Any sort of irregularities can reduce the accuracy rate significantly and data-preprocessing helps to normalize the strokes and remove any irregularities [26]. Lots of distortion can happen while collecting the data. Some of them are uneven text size, tremors, left-right bend, uneven spacing and points missing during pen movement. Data preprocessing works on those issues and makes it readable data for the model. Lots of pre-process methods were used on the collected data. Some of the important method descriptions are given below:

4.1 Segmentation

As we have mentioned, the data was collected in a form which consists of many characters and segmentation is the very first method of data pre-processing. Segmentation used to break down a multi-character input picture into separate characters. This segmentation also consists many sub-methods which are given below:

4.1.1 Form Scanning

After collecting the form, the very first step is to scan the form. After scanning the forms, the user can use it for any efficient usage of the data. There are two ways to scan the data. Either use Windows operating systems which take 1 min to scan and on the other hand Ubuntu with 600 dpi which take only 10-15 sec for scanning. For time's sake, Bangla Lekha Isolated team uses Ubuntu for scanning. A little snippet of scanned form is shown in figure 4.1.1.


[redacted] 01_0001_0_15_0916_0014

Form ID: [redacted]

Age: 15 Gender: Male District: Comilla Institution: [redacted]

অ	আ	ই	ঈ	উ	ঊ
ঋ	এ	ঐ	ও	ঔ	ক
খ	গ	ঘ	ঙ	চ	ছ
জ	ঝ	ঞ	ট	ঠ	ড

Figure 4.1.1: Scanned Form

4.1.2 Skew correction and crop

It was impossible to scan all the forms with the same angle. This is where the skew correction and cropping are used. By using clever edge detection, a large black border will aid in the discovery of the largest contour. In further steps it helps to crop all of the images in the same form, size, and angle by correcting the paper's skew [27].

4.1.3 Row wise cropping

To separate all the characters a row wise cropping is needed. A total 14 rows were there in the form and after cropping them the 14 pictures were saved in 14 different folders which were later used for further processing. Figure 4.1.2 shows the example of row wise cropping images.

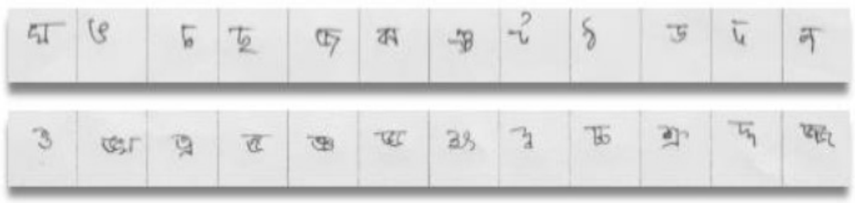


Figure 4.1.2: Example of Row Wise Cropping

4.1.4 Column wise cropping

Each row contains 6 characters. A column wise cropping is needed to perform after row wise cropping. This will separate all the 84 characters into different images which will be stored into 84 folders for future usage. Figure 4.1.3 shows the example of column wise cropping images.

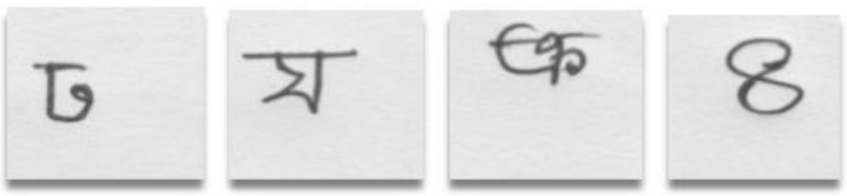


Figure 4.1.3: Example of Column Wise Cropping

4.2 Noise removal

A method known as noise removal is used to remove undesirable or undesired patterns. The function is commonly used to convert a grayscale picture to a bi-level (binary) image or filtering off pixels with too tiny or too big values is used to remove noise. In this dataset Otsu Algorithm is used to determine the optical threshold and for smoothing the image, gaussian blur was used. Figure 4.2.1 shows the example of noise removed images.

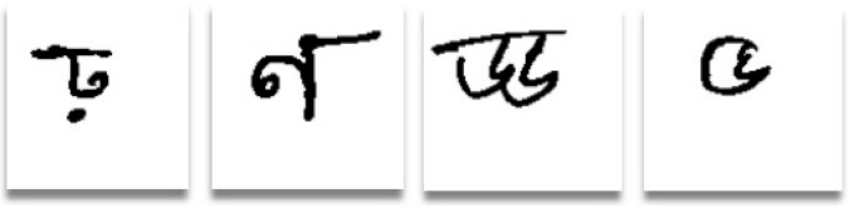


Figure 4.2.1: Example of Noise Removal Images

4.3 Resize

Image resizing was used to resize the image with a square shape. For the model we need to maintain the drawn character's aspect ratio. And by resizing we manage to provide the padding which was needed for the maintenance. Figure 4.3.1 shows the example of resized images

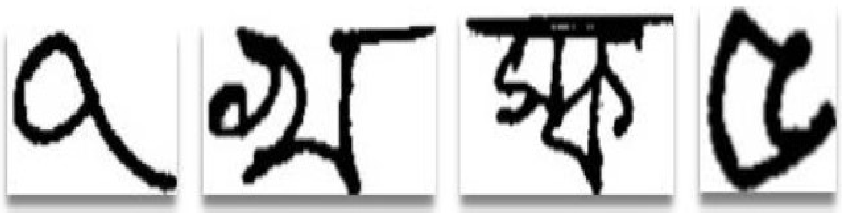


Figure 4.3.1: Example of Resized Image

4.4 Invert

After that, an inversion layer was needed to make the background black but as the character's color was black too the image had to make the brighter part dark and dark part too bright. To make this we have to change the pixel value. Figure 4.4.1 shows the example of inverted images.

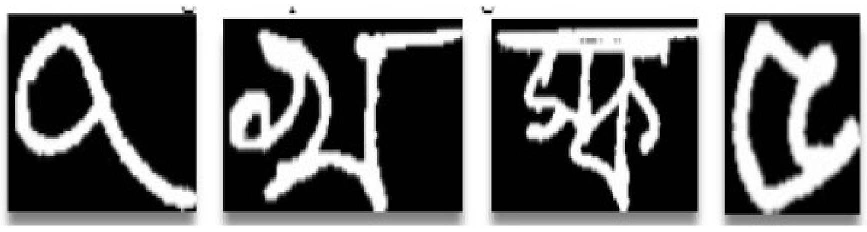


Figure 4.4.1: Example of Inverted Image

Chapter 5

Proposed Methodology

5.1 Architecture of Proposed Models

5.1.1 Proposed CNN model

CNN is mostly use for Image classification and it is one of the better than most of the other process because of its unique feature parameters sharing and dimension reduction system. To categorize photos into their assigned classes, we are proposing a multi-layered deep CNN model using the given approach. Most important of them is the Convolutional layer[28], fully connected layer [29] which connect all the neuron, and Pooling layer [30] are the three core layers of a CNN model. We employed two extra layers in addition to these three: the Activation layer and the Dropout layer. The following sections go through the specifics of each layer:

1. Convolutional Layer: The very first layer of CNN model is the Convolutional Layer. The main use of this layer is to extract information from the images. The convolution maintains the link between the pixels by learning visual qualities using small chunks of incoming data [31]. The two-input parameter are image input and a filter or a kernel. Edge detection, blurring, and sharpening may all be accomplished by applying convolution on a picture with many filters. For our custom model we have used 9 Conv2D layer.

2. Pooling Layer: Because of the huge number of parameters the model works so slow and to minimize the number we use the pooling layer. Our used dataset has different size of images. And some of them can be too large and pooling layer reduce the number of parameters from the images. In our proposed model we use MaxPooling2D to Calculate the maximum value for each feature map patch.

3. Fully Connected Layer: The structure is similar to Neural network. First it flattens the matrix into a vector and send it to the next fully connected layer. The following layers are included in the model:

    - Flatten layer: After the last MaxPolling2D, one flattens layer is utilized. This helps to spread the network as a whole.
    - Dense layer: Our proposed model consists four Dense layers. The main motive to use this layer is to feed the output with the neuron from the previous layer to the network.

- Dropout layer: There is a huge chance of overfitting the model. To avoid this problem, we normally use dropout layer.
- Activation layer: Lastly, we use SoftMax function for out multiclass classification. The following equation is used in the SoftMax function.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

In the equation Z is the input vector. Zi are the values of the input vector. Ezi is the standard exponential. The normalization term is the term at the bottom of the formula. And lastly K is the number of classes in the multi-class classifier.

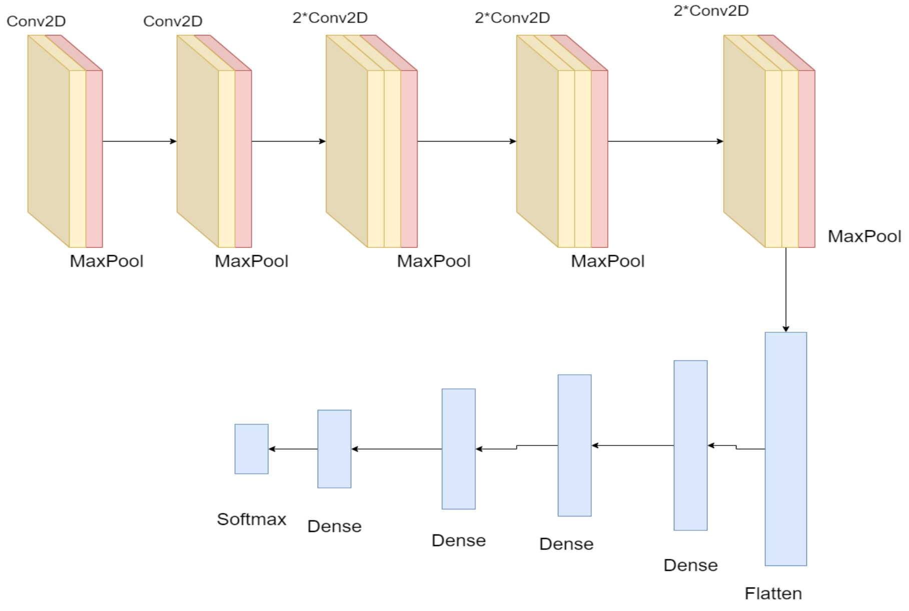
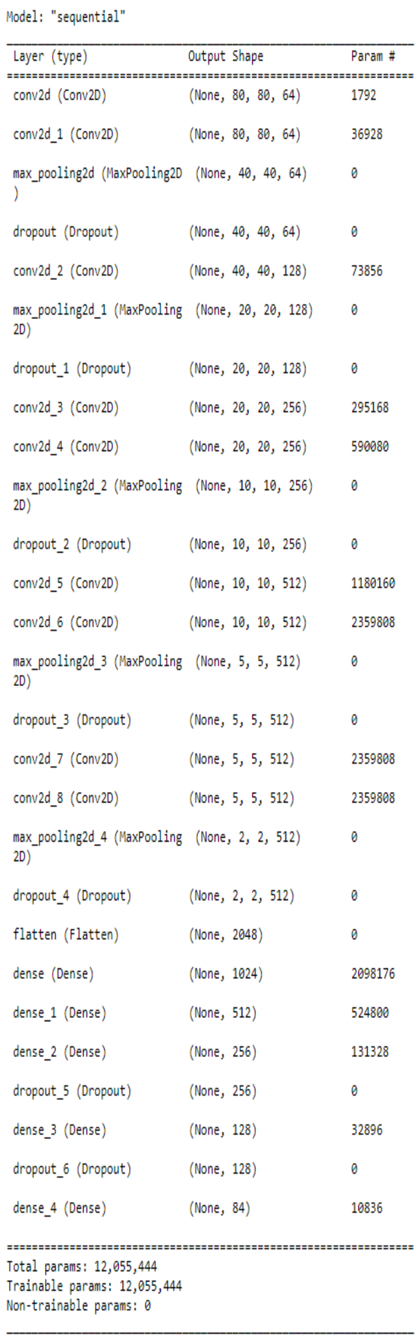


Figure 5.1.1: Proposed CNN model Architecture

Figure 5.1.1 shows the model Architecture of Our proposed CNN Model.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 80, 80, 64)        1792

 conv2d_1 (Conv2D)           (None, 80, 80, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 40, 40, 64)       0
 )

 dropout (Dropout)           (None, 40, 40, 64)        0

 conv2d_2 (Conv2D)           (None, 40, 40, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 dropout_1 (Dropout)         (None, 20, 20, 128)       0

 conv2d_3 (Conv2D)           (None, 20, 20, 256)       295168

 conv2d_4 (Conv2D)           (None, 20, 20, 256)       590080

 max_pooling2d_2 (MaxPooling  (None, 10, 10, 256)      0
 2D)

 dropout_2 (Dropout)         (None, 10, 10, 256)       0

 conv2d_5 (Conv2D)           (None, 10, 10, 512)       1180160

 conv2d_6 (Conv2D)           (None, 10, 10, 512)       2359808

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 512)        0
 2D)

 dropout_3 (Dropout)         (None, 5, 5, 512)         0

 conv2d_7 (Conv2D)           (None, 5, 5, 512)         2359808

 conv2d_8 (Conv2D)           (None, 5, 5, 512)         2359808

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 512)        0
 2D)

 dropout_4 (Dropout)         (None, 2, 2, 512)         0

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dense_1 (Dense)             (None, 512)               524800

 dense_2 (Dense)             (None, 256)               131328

 dropout_5 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dropout_6 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 84)                10836

=================================================================
Total params: 12,055,444
Trainable params: 12,055,444
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.1.2: Proposed CNN model Summary

Figure 5.1.2 shows the model Summary of our proposed model.

5.1.2 Pre-trained Models

Over the past few years lots of model has shown great improvement in CNN image classification. The neural network has been used in previous instances and has gathered data that may be applied to fresh targeted samples. We have used four pre-trained model for experiment. Those models are: VGG16, VGG19, Resnet18, Resnet34, InceptionV3. A very brief detailed of each model are given below: Numbered (ordered) lists are easy to create:

1. **VGG16:** This model one of best-known model for image classification and was Suggested in 2014 [32]. In the 2014 ILSVRC competition, this model took first and second place in the large-scale picture classification categories. Because of its 13 convolutional layers and 3 fully connected layers which are customizable it's called VGG16. Conv, Max-Pool, and Fc have 13,5 and 3 layers, respectively. With a parameter of 138 million and size of 500MB it is one of the largest networks in CNN [33]. In the first block it contains 64 filters which are subsequently doubled for the flowing blocks and ends with 512. 4096 neurons are being used in the first two FC and last layers its 1000 with softmax activation. The architecture is given below.

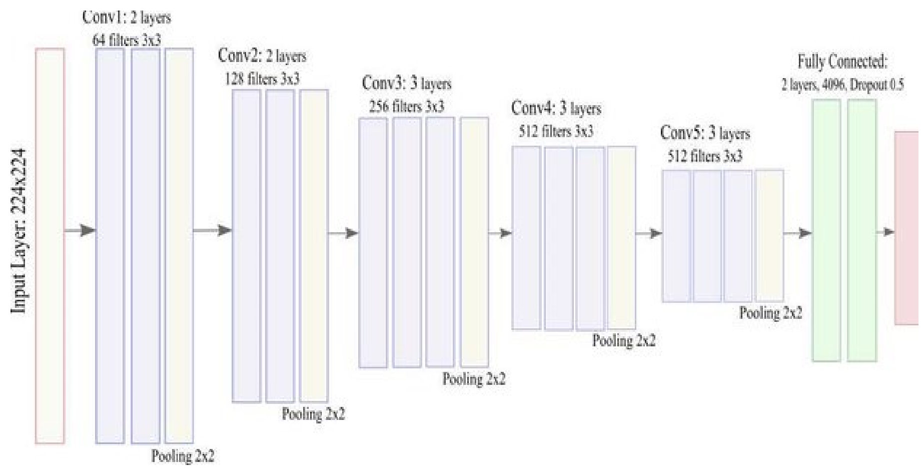
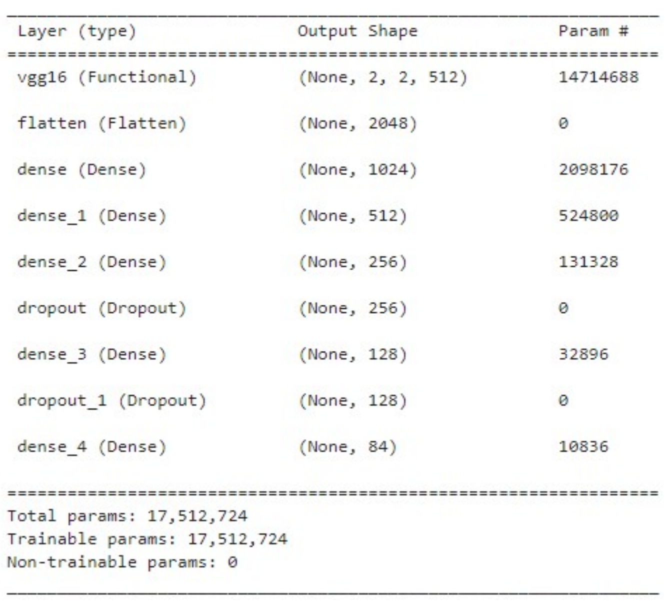


Figure 5.1.3: VGG-16 model Architecture

Figure 5.1.4 shows the summary of VGG-16


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 2, 2, 512)         14714688

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dense_1 (Dense)             (None, 512)               524800

 dense_2 (Dense)             (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 84)                10836

=================================================================
Total params: 17,512,724
Trainable params: 17,512,724
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.1.4: VGG-16 model Summary

2. **VGG19:** A little update version of VGG16 is VGG19. Only difference between vgg16 and vgg19 is instead of 13 convolutional layers it has 16 convolutional layers. It also takes 224*224 RGB image as input which make the input

shape (224,224,3)[34]. The added preprocessing feature is it subtract the RGB mean for each pixel and do it over the whole dataset. Strides are made up of kernels with a size of (3*3) and a size of one pixel to cover the complete visual notion. Maxpool was used (2*2) and stride of 2.

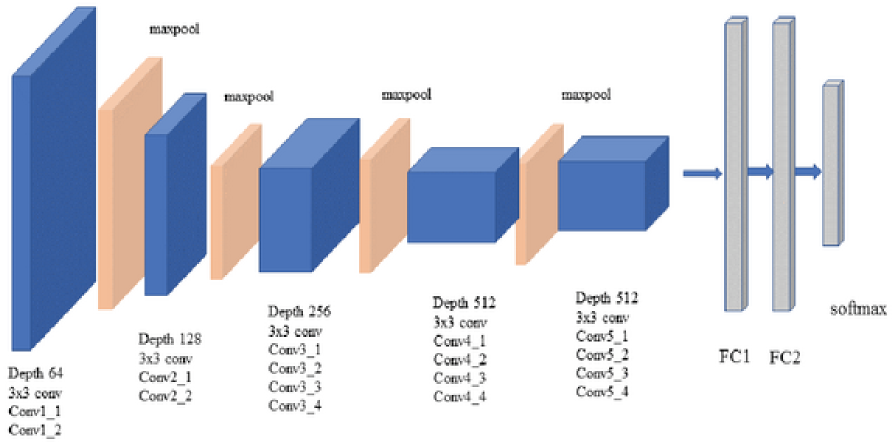
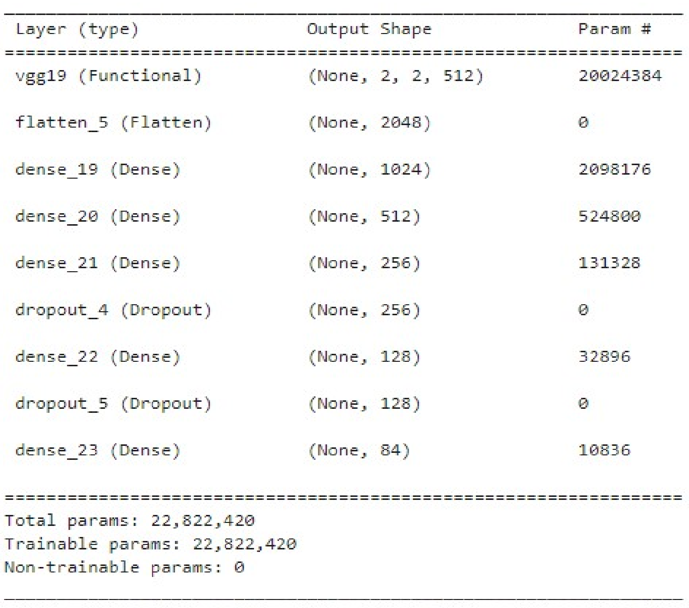


Figure 5.1.5: VGG-19 model Architecture

Figure 5.1.6 shows the summary of VGG-19



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 2, 2, 512)         20024384

 flatten_5 (Flatten)         (None, 2048)              0

 dense_19 (Dense)            (None, 1024)              2098176

 dense_20 (Dense)            (None, 512)               524800

 dense_21 (Dense)            (None, 256)               131328

 dropout_4 (Dropout)         (None, 256)               0

 dense_22 (Dense)            (None, 128)               32896

 dropout_5 (Dropout)         (None, 128)               0

 dense_23 (Dense)            (None, 84)                10836

=================================================================
Total params: 22,822,420
Trainable params: 22,822,420
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.1.6: VGG-19 model Summary

3. **InceptionV3:** InceptionV3 is another famous model for image classification [35].The below descriptions describe what are the steps been taken in one at a time in InceptionV3 [36]

   - Smaller Convolutions: Smaller convolution processes are substituted for larger convolution operations, resulting in substantially faster training. For example, two 3*3 filters have only 18 properties where only one 5*5 filters have 25 properties [37].
   - Asymmetric Convolutions: In this layer 3*3 convolution can be replaced by a 1*3 and 3*1. This way the number of parameters can be higher.
   - Auxiliary Classifier: Beside the main classifier, an auxiliary classifier is also used while training the network. The loss is included into the real network loss by a tiny CNN placed among layers. In InceptionV3, as a regularizer, the auxiliary classifier is used.
   - Reduce grid size: Pooling is one of the most common used processes to decrease the grid size. But a better and efficient process has been used to reduce the computational cost
   - Factorized Convolutions: In order to maximize the efficiency of computation, this layer reduce the parameters which are used. Beside this it observes the network efficiency.

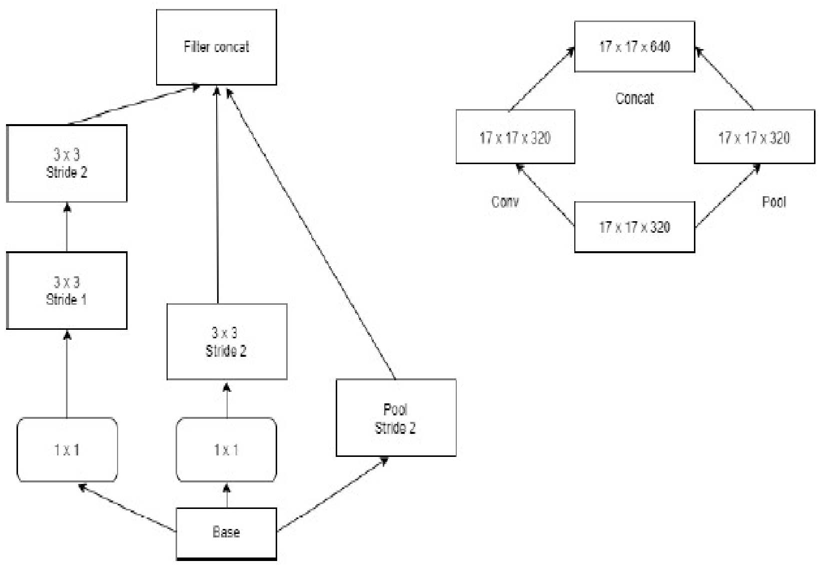


Figure 5.1.7: Reduce grid size architecture

Figure 5.1.8 shows the model Architecture of InceptionV3.

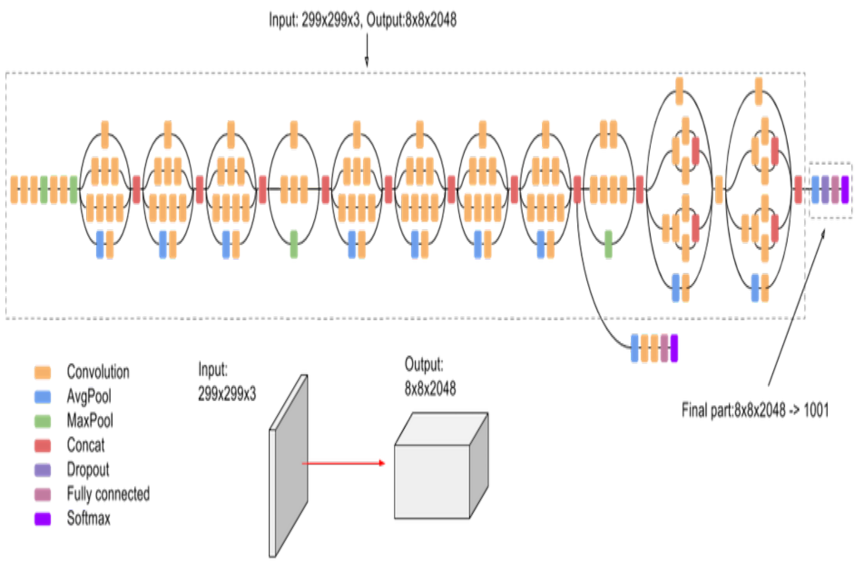
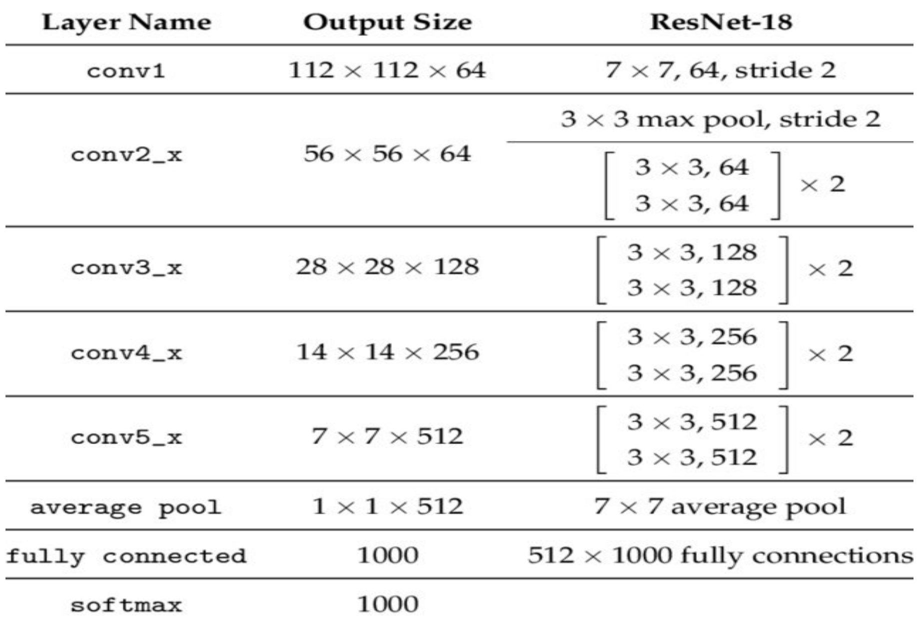


Figure 5.1.8: InceptionV3 model Architecture

4. **ResNet18:** Resnet is another CNN model that we have used on our dataset. Residual networks work well on image classification tasks. Based on the number of layers, different residual networks can be constructed. We have so far used one with 18 layers and one with 34 layers.

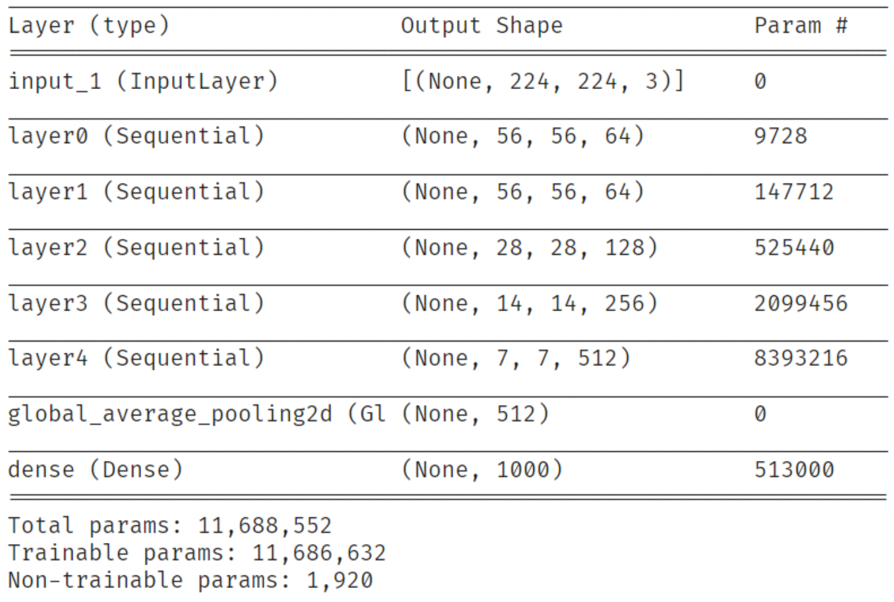
   Resnet is composed of many convolutions blocks. Each block is a basic CNN block with a Convolutional layer followed by batch normalization. Convolutional layer does convolutional operation on the input image. Batch normalization technique improves the training performance by making the output of each neuron closer to the mean. The output of the batch normalization is then fed towards the ReLu layer which is followed by a maxpool layer. The network is repeated in this configuration until the final layer which is a densely connected layer. The output layer has 84 distinct output classes each corresponding to a class in BanglaLekhaIsolated Dataset. In the case of resnet 18, there are 18 blocks thus the network has 18 as a suffix after the name. The whole network that we have implemented has 11,688,552 trainable parameters. The implementation of the Resnet 18 that we have used has the following architecture:



Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 5.1.9: Resnet18 model Architecture

Figure 5.1.9 shows the model Architecture of ResNet18.

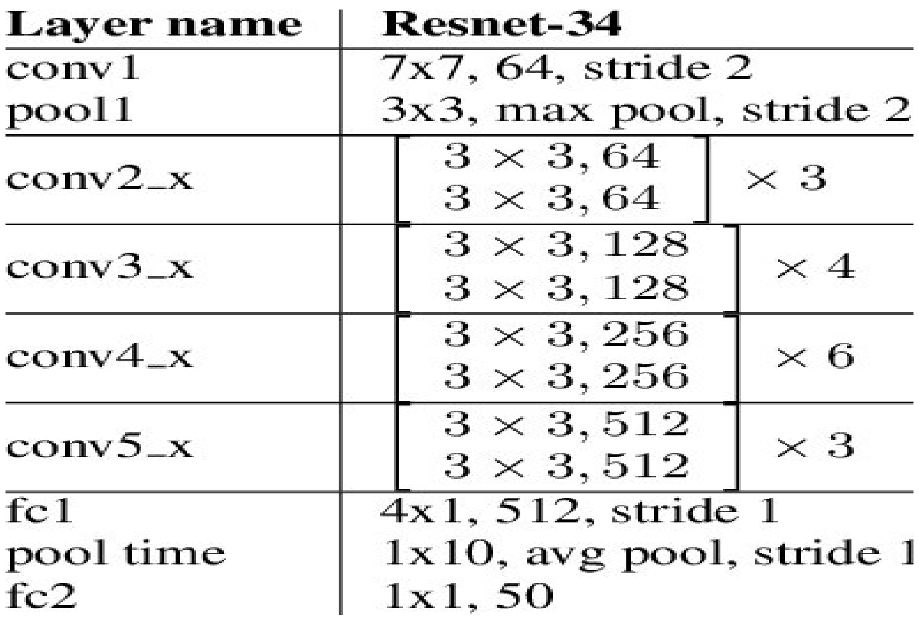


```
Layer (type)                 Output Shape              Param #
================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
________________________________________________________________
layer0 (Sequential)          (None, 56, 56, 64)        9728
________________________________________________________________
layer1 (Sequential)          (None, 56, 56, 64)        147712
________________________________________________________________
layer2 (Sequential)          (None, 28, 28, 128)       525440
________________________________________________________________
layer3 (Sequential)          (None, 14, 14, 256)       2099456
________________________________________________________________
layer4 (Sequential)          (None, 7, 7, 512)         8393216
________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
________________________________________________________________
dense (Dense)                (None, 1000)              513000
================================================================
Total params: 11,688,552
Trainable params: 11,686,632
Non-trainable params: 1,920
```

Figure 5.1.10: Resnet18 model Summary

Figure 5.1.10 shows the model Summary of ResNet18.

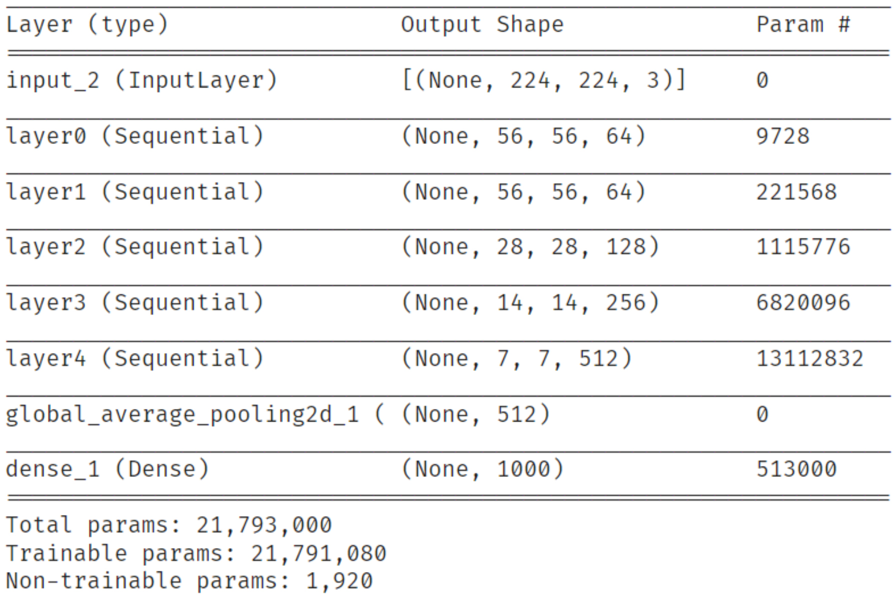
5. **ResNet34:** Resnet 34 is another variant of Residual networks which has 16 more layers in its architecture compared to resnet 18. Since it has more layers, the number of trainable parameters in the resnet 34 model is 21,793,000 compared to 11,688,552 of resnet 18. The inner building blocks of Resnet 34 are quite similar to that of ResNet 18. Each block has a convolutional 2d layer followed by batch normalization, ReLu activation and maxpool layer. After repeated structure, the network is then flattened to a Linear output layer consisting of 84 classes in total. The implementational structure of ResNet 34 is given below:



Layer name	Resnet-34
conv1 pool1	7x7, 64, stride 2 3x3, max pool, stride 2
conv2_x	$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3\times 3, 256 \\ 3\times 3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \end{bmatrix} \times 3$
fc1 pool time fc2	4x1, 512, stride 1 1x10, avg pool, stride 1 1x1, 50

Figure 5.1.11: Resnet34 model Architecture

Figure 5.1.11 shows the model Architecture of ResNet34.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
layer0 (Sequential)          (None, 56, 56, 64)        9728
_________________________________________________________________
layer1 (Sequential)          (None, 56, 56, 64)        221568
_________________________________________________________________
layer2 (Sequential)          (None, 28, 28, 128)       1115776
_________________________________________________________________
layer3 (Sequential)          (None, 14, 14, 256)       6820096
_________________________________________________________________
layer4 (Sequential)          (None, 7, 7, 512)         13112832
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              513000
=================================================================
Total params: 21,793,000
Trainable params: 21,791,080
Non-trainable params: 1,920
```

Figure 5.1.12: Resnet34 model Summary

Figure 5.1.12 shows the model Summary of ResNet34.

5.2 Used Parameters Values

Lots of parameters are to use to train a model. Among them most important parameters are epoch, batch size, regularization, callback, Learning rate, optimizer. Once the pre-processing is finished the images are ready to be used for the model. All the properties are changeable. Before the training is started, they all can be changed. Bangla lekha Isolation is one of the biggest datasets available right now. It has more 160K images divided into 84 classes. But for our research betterment we have randomly chose 1000 images from each class which make our dataset for the model 84000. We have divided those into three categories which are train, validation and test. Train and validation are used while training where test has been used later to calculate the accuracy score, precision, recall, f1 score. 50400 images are being used as training data. And validation and test both has 16800 images. We have used 30 epochs to train out model and the batch size was 32. Which make 1575 steps per epoch. “SGD” optimizer has been used. This is one of the popular optimizers along with “Adam”. Our proposed model is a little bit complex and its need a full efficient computation because of its huge output classes. And ”SGD” is perfect fit for the model as an optimizer. We initialize the learning rate as 0.001 and the minimum learnings rate was 0.0000001. A very high or very low learning rate can be problematic for the model. While updating the parameter a single record is used to remove the gradient decent problem. All our input images were RGB. The used Activation Function is “Softmax”. Because of the 84 output classes we had to use it. Other than this function another available function is “sigmoid” which is used to classify binary classes. For loss calculation we have used ”categorical crossentropy”. As there are 84 classes. And the output belongs to one of them and according to the output the loss is being calculated. We use “Google colab pro” for training the model. Where GPU were used which was assigned randomly by google [38]. We use “Model.fit” to start the training. Training and Validation accuracy, Training and validation loss were shown after every epoch. The following table 5.1 shows the values that were used in the process.

parameters	Proposed model	Pre-trained model
Training	60%	60%
Validation	20%	20%
Testing	20%	20%
Input size	(80,80)	(64,64)
Batch size	32	32
Epoch	30	30
Steps	1575	1575
verbose	1	1
learning rate	0.001	0.001
Minimum learning rate	0.000001	0.000001
Momentum	0.9	0.9
Environment	GPU	GPU
Optimizer	SGD	SGD
loss	categorical crossentropy	categorical crossentropy
Activation	Softmax	Softmax
Class mode	Multi class	Multi class
callback	ReduceLROnPlateau	ReduceLROnPlateau

Table 5.1: Used parameters for proposed and Pre-trained models.

Later test the model accuracy, precision, recall, f1-score was calculated[39],[40],[41]. The equation is given below:

- **Accuracy**[42]**:**

$$Accuracy = \frac{TruePositive + TrueNegative}{Truenegative + FalsePositive + TruePositive + FalseNegative} \tag{5.1}$$

- **Precision**[42]:

$$Precision := \frac{TruePositive}{TruePositive + FalsePositive} \tag{5.2}$$

- **Recall**[42]**:**

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5.3}$$

- **F1 Score**[42]**:**

$$F1score = \frac{2 * precision * recall}{Precision + Recall} \tag{5.4}$$

Chapter 6

Appraisal performance

6.1 Proposed Model Performance

As we have mentioned before, we have selected 200 random images from each class to test the model after the training. Beside this 60% was used to train the model and 20% was used for validation and 20% was unseen images for the model for later testing. After running the model, we got a very good training accuracy of 98.65%. Even for a simple model like our proposed model this a very promising accuracy. The following table 6.1 shows the accuracy and loss of the model.

Model	Training Accuracy	validation Accuracy	Training Loss	Validation Loss
CNN	98.55%	93.11%	0.09%	0.31%

Table 6.1: Training and validation Accuracy and Loss of our suggested Model.

According to the table, this proposed model gives a validation accuracy of 93.11%. The graph below shows the visual representation of the accuracy and loss of the model.

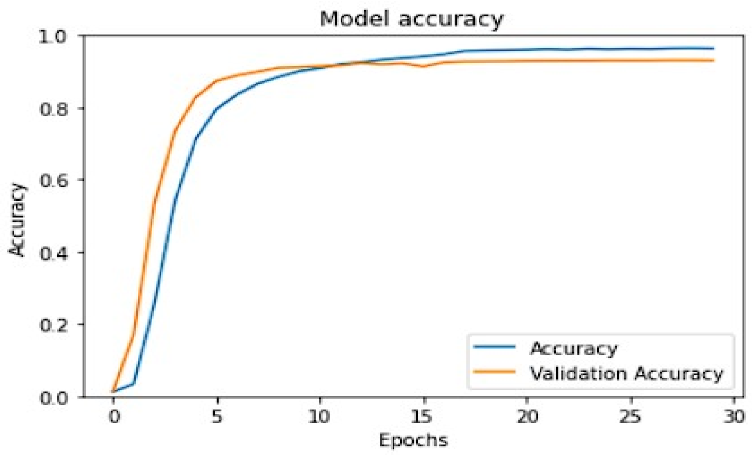


Figure 6.1.1: Suggested models training and validation accuracy

Figure 6.1.1 shows the models Training and Validation accuracy.

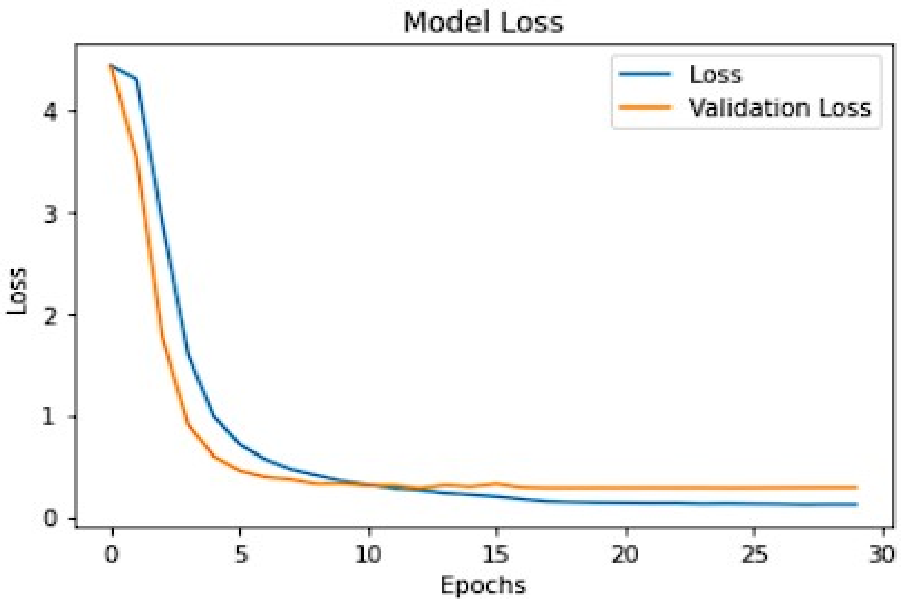


Figure 6.1.2: Suggested models training and validation loss

Figure 6.1.2 shows the models Training and Validation loss.

Moreover accuracy [43],[44], recall, precision, f1 score were calculated after the training[45]. Values are given below in table 6.2:

Model	Recall	Precision	F1 score	Accuracy
CNN	0.9311	0.9332	0.9313	0.9315

Table 6.2: Summary table of our suggested Model.

6.2 Pre-Trained model Performance

Unseen images set was used to testify the pre-trained model. After running the models, we got very good validation and testing accuracy from VGG16 and VGG19. Both has a higher training accuracy which is above 99% and validation accuracy of 94.24% and 94.07% respectively. And while testing on the unseen images set, we got a very impressive accuracy of 94.66% and 94.57% over 84 classes. Other than those two models, InceptionV3, ResNet18, ResNet34 gives decent testing accuracy with 91.97%, 90.24% and 88.39% respectively. General comparison of Training and testing accuracy among the models are shown below:

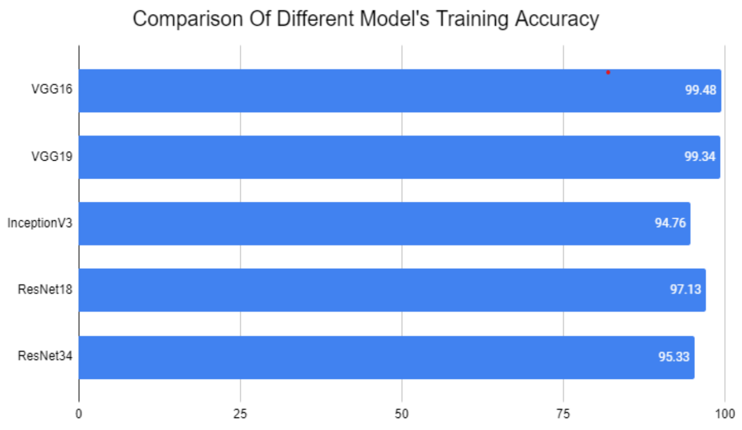


Figure 6.2.1: Pre-Trained models Training Accuracy

Figure 6.2.1 shows the Pre-trained models Training accuracy.

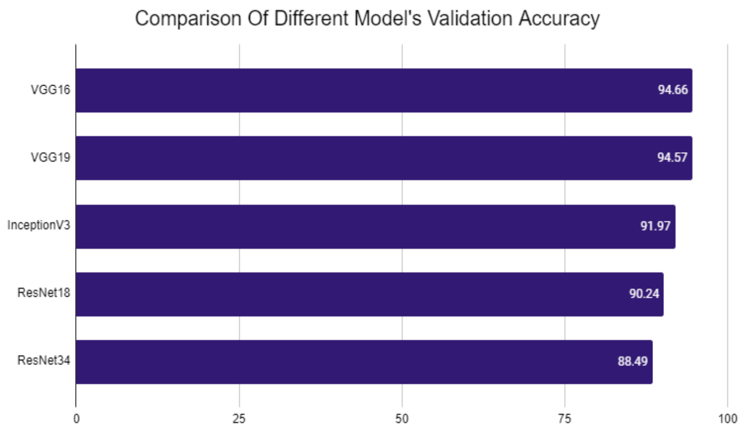


Figure 6.2.2: Pre-Trained models Testing Accuracy

Figure 6.2.2 shows the Pre-trained models Testing accuracy.

Further model evaluation information is given below. Where training and validation were done while training and testing was done on an unseen image set. So, the summary is the performance of the model on the testing images.

- **VGG16:** We got an accuracy of 99.48% on training and 94.67% on testing

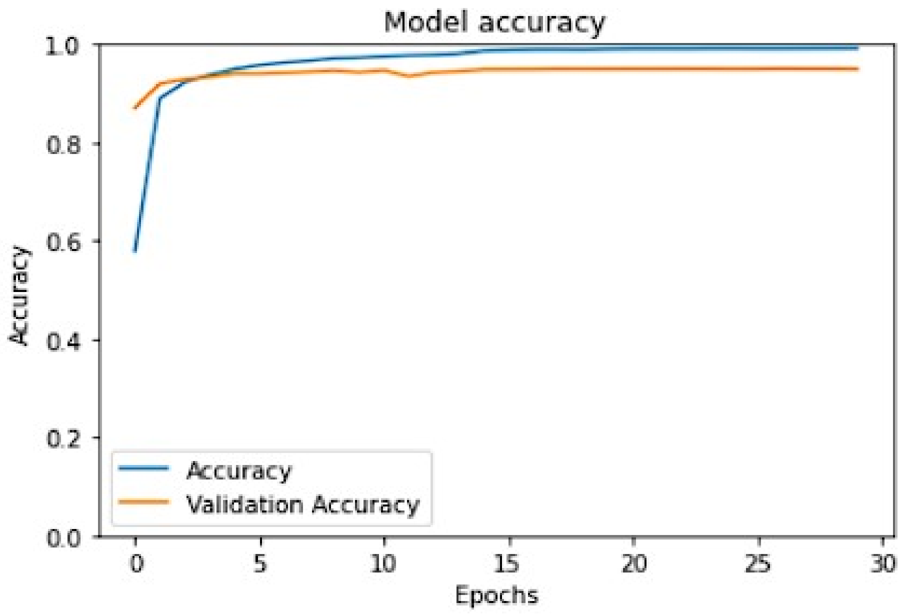


Figure 6.2.3: VGG16 models training and validation accuracy

Figure 6.2.3 shows the models Training and Validation accuracy.

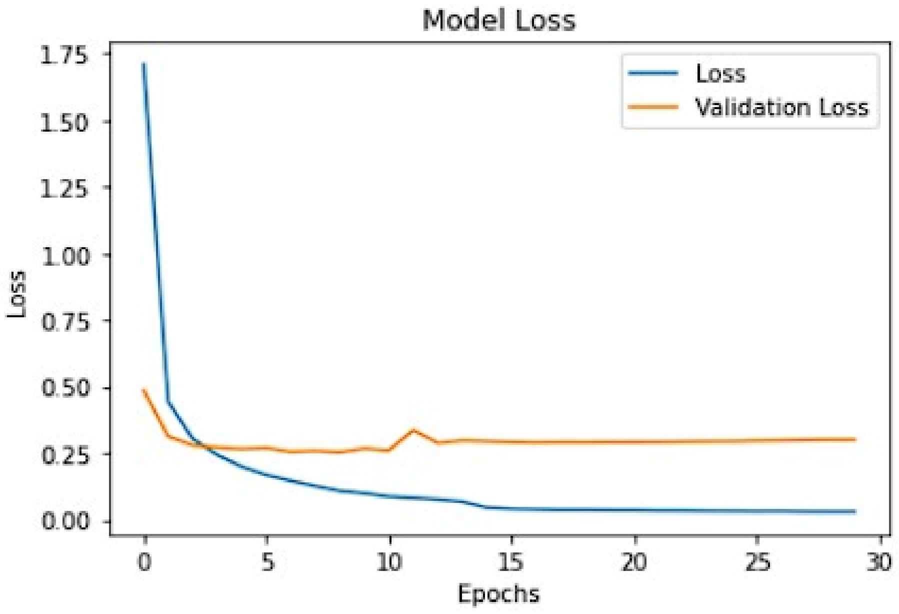


Figure 6.2.4: VGG16 models training and validation loss

Figure 6.2.4 shows the models Training and Validation loss.

- **VGG19:** We got an accuracy of 99.38% on training and 94.57% on testing

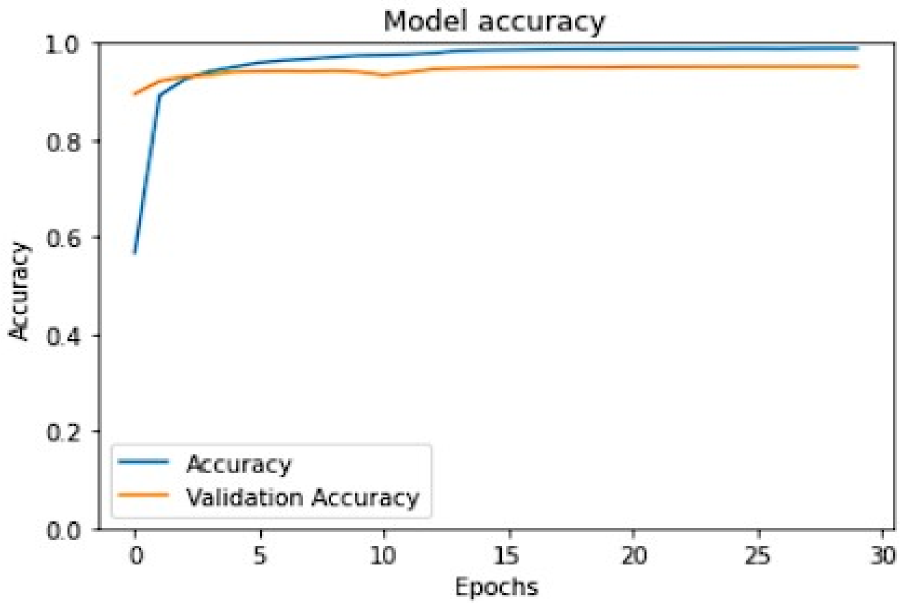


Figure 6.2.5: VGG19 models training and validation accuracy

Figure 6.2.5 shows the models Training and Validation accuracy.

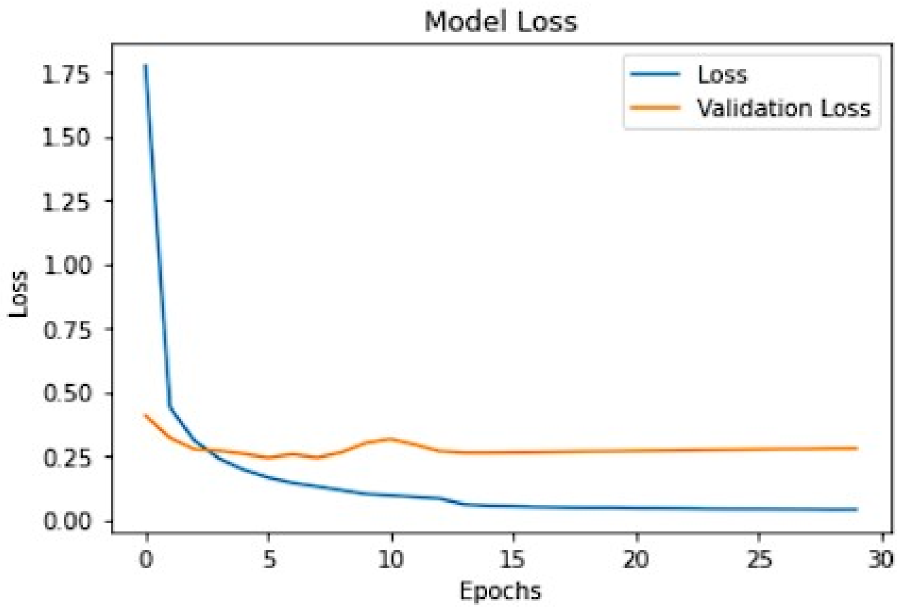


Figure 6.2.6: VGG19 models training and validation loss

Figure 6.2.6 shows the models Training and Validation loss.

- **InceptionV3:** We got an accuracy of 94.76% on training and 91.97% on testing

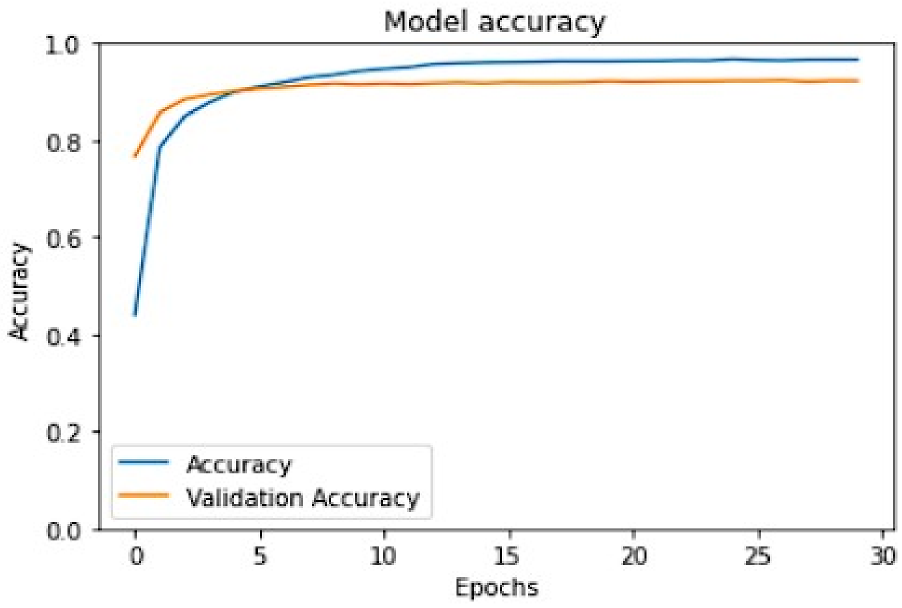


Figure 6.2.7: InceptionV3 models training and validation accuracy

Figure 6.2.7 shows the models Training and Validation accuracy.

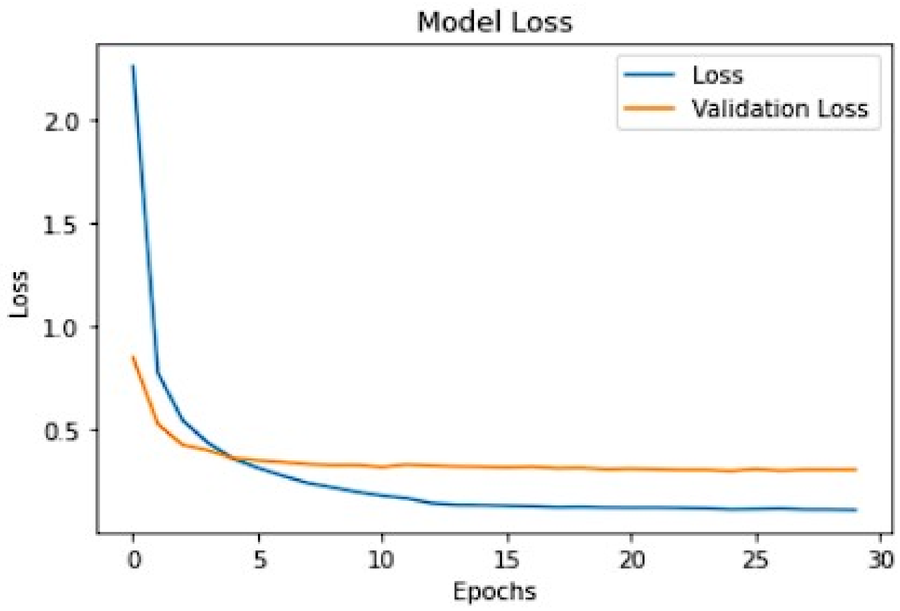


Figure 6.2.8: InceptionV3 models training and validation loss

Figure 6.2.8 shows the models Training and Validation loss.

- **ResNet18:** We got an accuracy of 97.13% on training and 90.24% on testing

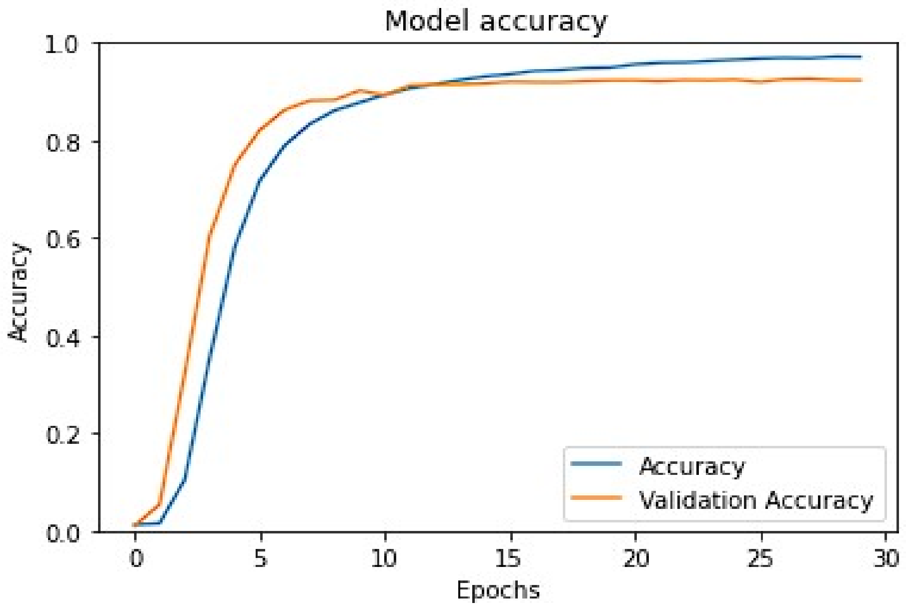


Figure 6.2.9: ResNet18 models training and validation accuracy

Figure 6.2.9 shows the models Training and Validation accuracy.

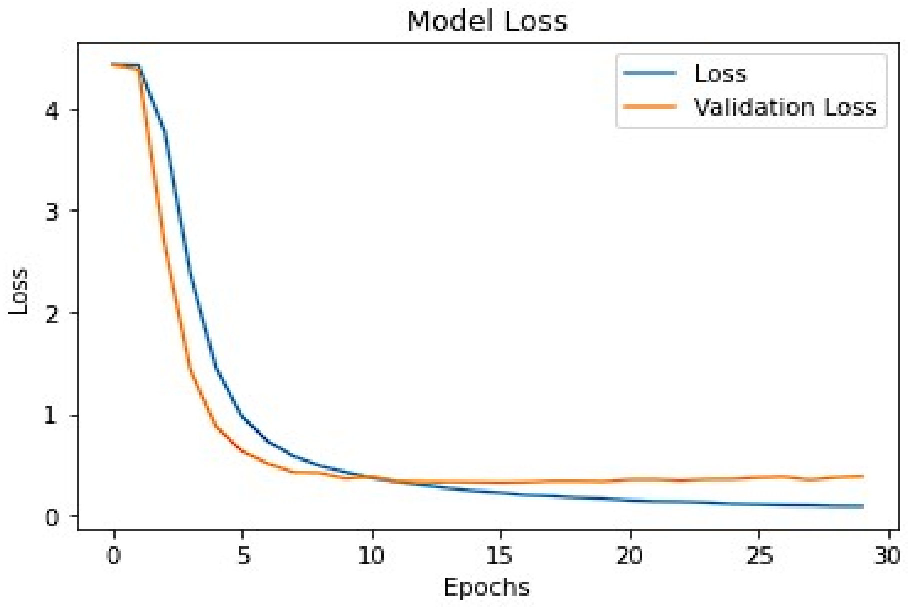


Figure 6.2.10: ResNet18 models training and validation loss

Figure 6.2.10 shows the models Training and Validation loss.

- **ResNet34:** We got an accuracy of 95.33% on training and 88.49% on testing

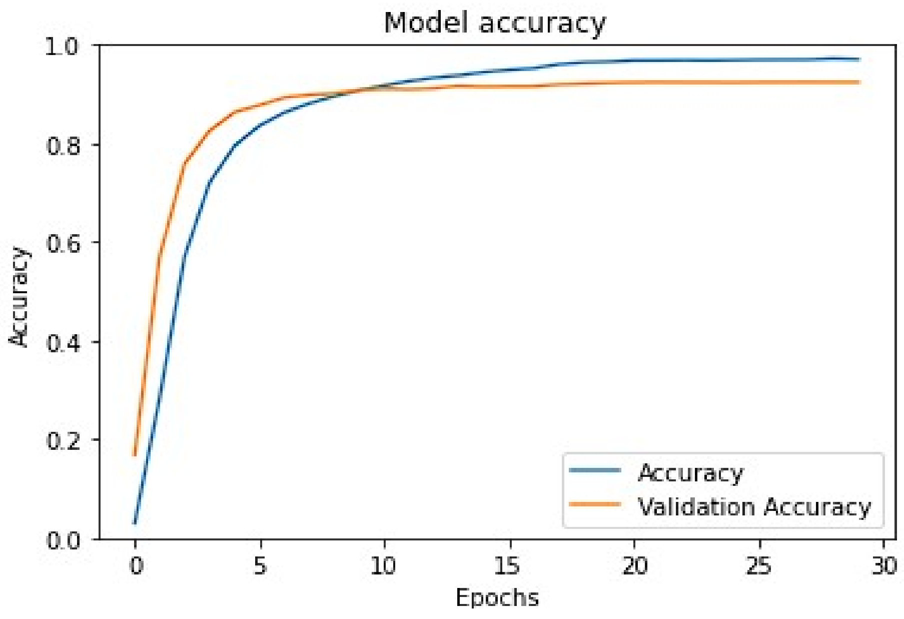


Figure 6.2.11: ResNet34 models training and validation accuracy

Figure 6.2.11 shows the models Training and Validation accuracy.

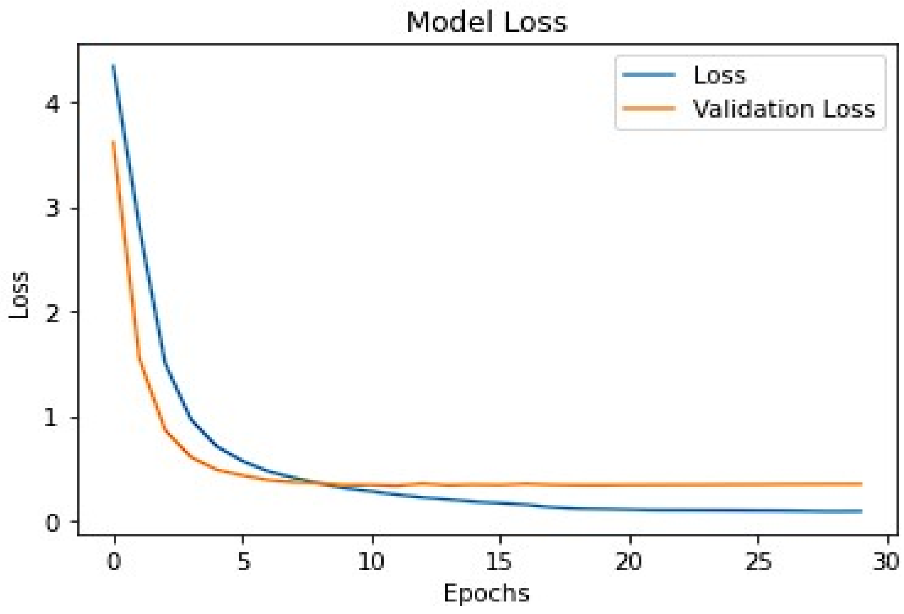


Figure 6.2.12: ResNet34 models training and validation loss

Figure 6.2.12 shows the models Training and Validation loss.

A summary table contain accuracy, precision, recall, f1 score for each model is given below:

Model	Recall	Precision	F1 score	Accuracy
Proposed CNN	**0.9311**	**0.9332**	**0.9313**	**0.9315**
VGG16	0.9453	0.9480	0.9462	0.9467
VGG19	0.9454	0.9474	0.9455	0.9457
InceptionV3	0.9239	0.9250	0.9229	0.9235
ResNet18	0.9020	0.9040	0.9018	0.9024
ResNet34	0.8815	0.8837	0.8706	0.8811

Table 6.3: Summary table of pre-trained model

6.3 Compare and Analysis

To compare our proposed model to other popular existing image classification models, we have tabulated all the metrics in Table 6.3. Our proposed model has achieved a recall score of 0.9311 out of 1. Recall is the number of True Positive predictions divided by the summation of True Positive and False Negative. Simply put, out of all of the possible true predictions how many a model has gotten right is the recall score. The higher the recall the better the model is. From the chart we can see that our proposed model achieves a better recall value compared to resnet 18 , 34 and Inception V3 but is about 1% behind from VGG16 and VGG19. We can call this a satisfactory recall score.
On the other hand, precision is measured by taking the number of predictions done correctly out of all predictions. It is defined by the True Positive divided by the summation of True Positive and False Positive. Similar to recall score, higher precision also indicates a better model. Our proposed model has a precision score of 0.9332 which is higher than both resnet 18, 34 and Inception V3 but slightly lower than VGG 16 and 19. The difference between our model and VGG 16 and 19 are really slim.
Additionally, F1 score measures the harmonic mean of both precision and recall. Simply put, F1 score balances both precision and recall and gives one value to measure both and compare different models. In the case of F1, a similar trend is found. VGG16 and 19 performs better than our proposed model but Resnet 18 and 34 and Inception V3 achieves a lower F1 score than our model.
Accuracy is the simplest of metrics that measures how many of the predictions are accurate out of all predictions. In this test, our model again shows a similar kind of performance trait, that is , it is better than Resnet, Inception V3 but slightly lower than VGG 16 and 19.
Overall, our model achieves almost a similar level of accuracy of VGG 16 and 19 while using far less number of parameters which means, our model will be much faster while training while maintaining a similar level of performance. This would be useful in cases where the dataset is large in size and the computation environment is very costly. Using our model in this case would both save time and money compared to other models.

Model	Parameters	Time per Epoch	Final Accuracy
VGG16	17,512,724	77 Sec	0.9466
VGG19	22,822,420	89 Sec	0.9457
InceptionV3	23,851,784	118 sec	0.9197
ResNet18	11,688,552	57 Sec	0.9024
ResNet34	21,793,000	76 Sec	0.8849
Proposed CNN	**12,055,444**	**63 Sec**	**0.9315**

Table 6.4: Time and Cost comparison

The table 6.4 shows the number of parameters and time has taken per epoch for all the models. As we can see, comparing to the large size model like VGG16,VGG19 and ResNet34 which has a large number of parameters, our proposed model has a very small amount of parameters. The amount of time that has taken to train the model is far small than the pre-trained model. And this can be useful to train large size dataset. If we want to save our time and computational cost maintaining a good accuracy then this model is better than any of the pre-trained model.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We have proposed our own custom CNN model for Bengali Handwritten digit classification. Our proposed work has provided some great results. Along with other excellent works of researchers working in Bengali handwritten text classification, our work will contribute in this field as an additional option in choosing an appropriate model for use in everyday life tasks. We wish to contribute to the use of Bengali language in computing through this paper.

7.2 Future Work

We have so far implemented our very first model and have gained experience using other popular CNN models. We have achieved satisfactory results but there is more room for improvement in many areas. We would like to work on text scanning and line separation with automated word classification and correction in future. That future model will be based on this work. If we succeed, we will be able to create an automated Bengali handwritten text scanner that will convert handwritten documents in digital formats.

Bibliography

[1] A. K. Ray and B. Chatterjee, "Design of a nearest neighbour classifier system for bengali character recognition," *IETE Journal of Research*, vol. 30, no. 6, pp. 226–229, 1984.

[2] A. Dutta and S. Chaudhury, "Bengali alpha-numeric character recognition using curvature features," *Pattern Recognition*, vol. 26, no. 12, pp. 1757–1770, 1993.

[3] U. Pal and B. B. Chaudhuri, "Indian script character recognition: A survey," *Pattern Recognition*, vol. 37, no. 9, pp. 1887–1899, 2004.

[4] T. K. Bhowmik, U. Bhattacharya, and S. K. Parui, "Recognition of bangla handwritten characters using an mlp classifier based on stroke features," *International Conference on Neural Information Processing*, pp. 814–819, 2004.

[5] A. S. Mashiyat, A. S. Mehadi, and K. H. Talukder, "Bangla offline handwritten character recognition using superimposed matrices," *Proc. 7th International Conf. on Computer and Information Technology*, pp. 610–614, 2004.

[6] U. Bhattacharya, M. Shridhar, and S. H. Rarui, "On recognition of handwritten bangla characters," *ICVGIP*, pp. 817–828, 2006.

[7] A. A. Chowdhury, E. Ahmed, S. Ahmed, S. Hossain, and C. M. Rahman, "Optical character recognition of bangla characters using neural network: A better approach," *2nd ICEE*, 2002.

[8] U. Pal and B. B. Chaudhuri, "Automatic recognition of unconstrained offline bangla handwritten numerals," *Advances in Multimodal Interfaces ICMI 2000*, pp. 371–378, 2000.

[9] A. R. M. Forkan, S. Saha, M. M. Rahman, and abdul sattar, "Recognition of conjunctive bangla characters by artificial neural network," *Information and Communication Technology, 2007*, pp. 96–99, 2007.

[10] M. M. Rahman, M. A. H. Akhand, S. Islam, P. C. Shill, and M. M. H. Rahman, "Bangla handwritten character recognition using convolutional neural network," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 7, no. 8, pp. 42–49, 2015.

[11] S. Chatterjee, R. K. Dutta, D. Ganguly, K. Chatterjee, and S. Roy, "Bangla handwritten character recognition using convolutional neural network," *Computer Vision and Pattern Recognition*, pp. 42–49, 2018.

[12] M. A. Hasan, S. Ahmed, and M. A. R. Alif, "Isolated bangla handwritten character recognition with convolutional neural network," *20th International Conference of Computer and Information Technology (ICCIT)*, 2017.

[13] H. Begum, A. Rafid, and M. M. Islam, "Recognition of bangla handwritten characters using feature combinations," *IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2018.

[14] S. Afroge, B. Ahmed, and D. M. A. Hossain, "Bangla optical character recognition through segmentation using curvature distance and multilayer perceptron algorithm," *Electrical, Computer and Communication Engineering (ECCE)*, pp. 253–257, 2017.

[15] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application," *Applied Soft Computing*, vol. 12, no. 5, pp. 1592–1606, 2012.

[16] M. Al-Amin, M. S. Islam, and S. D. Uzzal, "Sentiment analysis of bengali comments with word2vec and sentiment information of words," *Electrical, Computer and Communication Engineering (ECCE)*, pp. 253–257, 2017.

[17] M. N. H. Salim, M. M. Islam, N. A. Nipa, and M. M. Akbar, "A comparative overview of classification algorithm for bangla handwritten digit recognition," *International Joint Conference on Computational Intelligence*, pp. 265–277, 2020.

[18] C.-L. Liu and C. Y. Suen, "A new benchmark on the recognition of handwritten bangla and farsi numeral characters," *Pattern Recognition*, vol. 42, no. 12, pp. 3287–3295, 2009.

[19] M. Z. Alom, P. Sidike, T. M. Taha, and V. K. Asari, *Handwritten bangla digit recognition using deep learning*, 2017. arXiv: 1705.02680 `[cs.LG]`.

[20] S. Saha and N. S. Puja, "A lightning fast approach to classify bangla handwritten characters and numerals using a newly structured deep neural network," *Procedia computer science 132*, pp. 1760–1770, 2018.

[21] A. S. A. Rabby, S. Haque, S. Abujar, and S. A. Hossain, "Ekushnet: Using convolutional neural network for bangla handwritten recognition," *Procedia computer science 132*, pp. 603–610, 2018.

[22] L. Wan, M. D. Zeiler, S. Zhang, and R. F. Yann Lecun, "Regularization of neural networks using dropconnect," *International conference on machine learning*, pp. 1058–1066, 2013.

[23] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," *ESANN*, vol. 1, pp. 2–10, 2011.

[24] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," *ICCITECHN*, pp. 1058–1066, 2017.

[25] M. Nabeel, M. Sifat, A. Anowarul, *et al.*, "Banglalekha-isolated," 2017.

[26] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[27] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[28] B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” *Proceedings of the 4th ACM workshop on information hiding and multimedia security*, pp. 5–10, 2016.

[29] D. Sun, J. Wulff, E. B. Sudderth, H. Pfister, and M. J. Black, “A fully-connected layered model of foreground and background flow,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2451–2458, 2013.

[30] S. Albawi, T. A. Mohammed, and S. ALZAWI, “Understanding of a convolutional neural network,” *2017 International Conference on Engineering and Technology (ICET), Ieee*, pp. 1–6, 2017.

[31] W. Jiang, K. Zhang, N. Wang, and M. Yu, “Meshcut data augmentation for deep learning in computer vision,” *PLoS One*, vol. 15, no. 12, 2020.

[32] K. Simonyan and A. Zisserman, *Very deep convolutional networks for largescale image recognition*, 2014. arXiv: 1409.1556 `[cs.LG]`.

[33] M. Lin, Q. Chen, and S. Yan, *Network in network*, 2013. arXiv: 1312.4400 `[cs.LG]`.

[34] M. Mateen, J. Wen, Nasrullah, S. Song, and Z. Huang, “Fundus image classification using vgg-19 architecture with pca and svd,” *Symmetry*, vol. 11, no. 1, p. 1, 2019.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[37] M. Jafari, S. Samavi, N. K. S. Soroushmehr, K. R. Ward, and K. Najarian, “Automatic detection of melanoma using broad extraction of features from digital images,” *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE*, pp. 1357–1360, 2016.

[38] U. Yaqub and M. Al-Mouhamed, “Simulation acceleration of index modulation using a cluster of nvidia tesla k-80,” 2017.

[39] J. Myerson, L. Green, and M. Warusawitharana, “Area under the curve as a measure of discounting,” *Journal of the experimental analysis of behavior*, vol. 76, no. 2, pp. 235–243, 2001.

[40] B. G.Marcot, “Metrics for evaluating performance and uncertainty of bayesian network models,” *Ecological modelling*, vol. 230, pp. 50–62, 2012.

[41] M. Hossin and S. M.N, “A review on evaluation metrics for data classifcation evaluations,” *International journal of data mining knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[42] E. Hussain, M. Hasan, S. Z. Hassan, T. H. Azmi, M. A. Rahman, and M. Z. Parvez, “Deep learning based binary classification for alzheimer’s disease detection using brain mri images,” *15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1115–1120, 2020.

[43] F. F., “Testing accuracy,” *Forest Science*, vol. 6, no. 2, pp. 139–45, 1960.

[44] P. E. D, “A comparison of rater training programs: Error training and accuracy training,” *Applied Psychology*, vol. 69, no. 4, p. 581, 1984.

[45] M. Buckland and F. Gey, “The relationship between recall and precision,” *Journal of the American society for information science*, vol. 45, no. 1, pp. 12–19, 1994.