KDANet:
Optical Recognition for Bangla Language using Deep Neural Networks

by

Kazi Kamruzzaman Rabbi
18101625
Pranto Dev
18101424
Akram Hossain
18101416
Aninda Sadman
22141052

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

Department of Computer Science and Engineering
BRAC University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/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:

Kazi Kamruzzaman Rabbi
18101625

Pranto Dev
18101424

Akram Hossain
18101416

Aninda Sadman
22141052

Approval

The thesis/project titled "KDANet: Optical Recognition System for Bangla Language Using Deep Learning" submitted by

1. Kazi Kamruzzaman Rabbi(18101625)
2. Pranto Dev(18101424)
3. Akram Hossain(18101416)
4. Aninda Sadman(22141052)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 24, 2022.

Examining Committee:

Supervisor:
(Member)

Mr. Annajiat Alim Rasel
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dewan Ziaul Karim
Lecturer
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

When images of printed or handwritten are converted; be it mechanically or electronically to an editable text format, this is called optical character recognition. Bangla is one of the most complex languages as it has so many characters and digits. Moreover the Bangla language has about 300 composite characters. That is why the extraction of characters from images is more difficult for Bangla compared to other languages. Deep learning has recently developed good capabilities for extracting high-level features from an image kernel.

This paper will propose a custom model KDANet and compare with some popular deep learning models that can recognize handwritten Bangla characters written in various and distinct handwriting styles. These systems learn more accurate and inclusive features from large-scale training datasets than earlier feature extraction techniques.

Keywords: Character Recognition; Bangla OCR; KDANet; Computer Vision; Deep Learning; Convolutional Neural Networks;

Dedication

To our beloved parents and those who believed in us.

Acknowledgement

We are thankful for the utmost contribution from our university and respected faculties.

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

CV Computer Vision

DL Deep Learning

DNN Deep Neural Network

$LSTM$ Long Short Term Memory

RNN Recurrent Neural Network

Chapter 1

Introduction

Character recognition from images, especially handwritten image samples have received a lot of attention recently. Over these years, computation power has increased significantly due to high-performing computing systems. This actually paved the way for more and more complex neural network models which were limited a few years ago. It made the raw processing power demanding research easier. Many researchers have worked on character recognition over the last few decades, but research on Bangla character recognition is not so saturated yet, although it has been getting a lot of attention recently.

The Bangla script came from the ancient Brahmi script with numerous transformations. Bangla is the world's seventh most widely spoken language. Approximately 265 million people speak this language. It is enriched with many different characters. The Bangla language has 11 vowels, 39 consonants, and 10 numbers. There are also some compound and some special characters in the Bangla language.

Vowel	অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ
Constant	ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ
Compound Characters	ঙ্গ ঙ্গ ত্ত ঞ্চ ঞ্ছ ঞ্জ ষ্ণ ষ্মা ণ্ড ক্ষ
Special Characters	মৌ ন্বী চাঁ য্য ক্রো

Figure 1.1: Examples of Bangla Characters

Character detection has a greater implementation in terms of word detection and similarly in case of sentence level detection. An OCR system can be used to retain the structure of old documents. Over the years, many techniques have been developed and used to convert handwritten Bangla characters into digital data i.e Water overflow model, CNN, RNN, DNN, Tesseract, etc. The compound characters in the Bangla language are a bit difficult to recognize due to size, cursiveness, and miscellaneous nature of handwriting. Compound characters and the cursive nature of the Bangla language act as the main barrier for recognizing the Bangla language fully.

In [1] they used 24 compound characters and achieved 91.1% accuracy using CNN based architecture Inception. And in [16] they used shape decomposition technique along with chain code histogram and multi-layer perceptron. They used 128 compound characters, which cover 95% of the total compound characters used in the conventional Bangla language. They achieved 88.7% accuracy.

1.1 Research Problem

The Bangla language is a highly sophisticated language. There are lots of factors that make it difficult to distinguish characters. In addition to that, if we consider the case of handwritten Bangla texts, the handwriting varies greatly from person to person. Thus, in very few cases, it becomes almost impossible to understand clearly with the human eye, let alone machines. Therefore, we need to overcome many challenges but still, we thrive to propose a better OCR in terms of flexibility. Not to mention the many different edge cases we are deemed to face whilst trying to determine the suitable parameters for our aspiring model.

One of the most complicated things about the Bengali language is the header stroke (Matra), which differs from character to character and can change the meaning of a word dramatically. There are some rare cases where the characters differ only in header strokes. For example, ন and ণ are very close to each other and can only be differentiated by their header strokes (Matra). There are characters, which can be differentiated with only a dot. For instance, ব and র, these two characters are different only because of a dot, which makes it strenuous to distinguish. The vowels have modified versions too, [Fig:2], which combine with the consonants and give it a meaning. For example, the vowel উ is also used as ু with some characters and consonants like ঁ (Chandrabindu) is used to compliment other characters. In the word, "বাকা" here both ব and া are consonants, yet they complement each other in forming one letter. These make the character recognition a lot more challenging because we have to take care of these special 6 signs in consideration and some modified characters are similar to each other that makes our classification more complicated.

Vowel	Modified Vowels
অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ	◌া ◌ি ◌ী ◌ু ◌ূ ◌ৃ ◌ে ◌ৈ ◌ো ◌ৌ

Figure 1.2: Examples of Bangla Vowels

1.2 Research Objective

If we think about what a standard OCR should be, the worldwide standard of English is phenomenal. Compared to that, there is very little progress done for our Bangla language. Our primary objective is to build a system which can convert any image filled with Bangla texts into an editable text format. For that, we obviously need a hybrid model architecture so we need to conduct extensive research on what our fellow researchers have achieved so far on Bangla OCR.

To build our custom model, we need to understand deeply how the state-of-the-art architectures are functioning and which hyperparameter tuning shows promise. In this regard, we can take reference from English OCR engines and test ourselves on Bangla popular datasets. As such, we can grasp how the different models are making predictions on our dataset and detect the edge cases on which we need to work on. After we are done with the trial and error mechanism, we can combine the positives and tune the hyperparameters to tackle the negatives which will most likely lead to our ultimate hybrid architecture.

We might need to implement transfer learning from one model to another for dealing with varied issues. For instance, a BLSTM can be added on the post-processing to handle sequences of characters or words or even sentences from data. Additionally, we hope to create a synthetic dataset with a mixture of both printed and handwritten Bangla writings. Since printed dataset is hardly available, our plan is to make use of the existing fonts for Bangla language. Our goal is to upgrade the Bangla language OCR system towards the widely used English standard OCR engines in near future with our research.

Chapter 2

Literature Review

2.1 Optical Character Recognition

As human civilization is moving forward, the necessity for old-scripted documents is increasing. To take inspiration from age-old methods or to know about old culture, these scripts need to be preserved in digital format. Optical Character Recognition is the technology that helps to recognize handwritten text and convert them into text.

2.2 Optical Character Recognition Techniques

There are a few fundamental steps involved in building OCR regardless of which method is used. We will have a quick overview of these steps[15] and some techniques.

2.3 Scanning

Scanning is the conversion of physical documents into a digital format. This conversion process is called text digitization. Scanning can be simply done by taking images of documents or using a scanner to scan documents. Depending on the scanning process, the resolution varies, performance also depends on the resolution or quality of the digital format.

2.4 Pre Processing

Before feeding data into the feature extraction layer we need to process the data. Few novel techniques:
Image data augmentation: To introduce the variety in training, techniques like data resizing, rescale,wrap, etc are used.
Image grayscale: In order to reduce dimensionality of data and reduce overhead for feature extraction layer, image grayscaling is a great tool. It converts RGB(3D) images into grayscale(1D).
Image noise reduction: Converting RGB images into grayscale introduces noise. Depending on the dataset we perform noise reduction, specifically for languages like Bangla where their character includes dot, noise reduction can be troublesome. Because there are Bangla characters containing dot(.) like : 'র', this dot can be misinterpreted as noise data.
Normalization: [9] The pixel intensity value of images usually varies from 1 to 255. In order to convert this value from 0 to 1 (normal distribution helps visualize the image better), we use the image normalization technique. It is done by dividing the matrix representation of image by 255.

2.5 Feature Extraction

Feature extraction is a technique by which we can extract the most prominent features from a text image. It is used to classify the characters according to the relevance of the extracted features. Later this feature extraction becomes of great help during the classification phase.

2.6 Classification

Classification is the phase where character recognition takes place. In this process, the machine learning algorithm chooses the appropriate space to which the extracted features belong and gives a prediction.

2.7 Related Works

In [11] the authors have proposed a system for the recognition of compound characters in the Bangla language using a Deep Neural Network with a squeeze and excitation ResNeXt (SE-ResNeXt) model. The proposed model has several SE-ResneXt blocks followed by a convolutional block. The Squeeze-and-Excitation Block is an architectural block that allows dynamic channel-wise feature recalibration to increase its recognition performance. In this block, they have used global average pooling to squeeze each channel to a specific value. Then followed by a fully connected layer, they have used the RELU activation function to achieve nonlinearity and non-mutually exclusive relationships. Each channel has a smooth gating function using another dense layer followed by a sigmoid. To deal with overfitting, they used the dropout technique, which will remove some neurons randomly during the training phase and that made the model more generalized. Lastly, they scaled the output. This proposed model has an accuracy of 99.82% and F1 score of 97.62% for recognizing compound characters in the Bangla language.

The authors[10] proposed a technique for the recognition of printed Bangla characters using a multi-layer feed-forward back propagation Neural Network. They created an image dataset of each 40x40 pixel and used a multi-layer perceptron classifier with one and two hidden layers in scikit-learn. The log-loss function is optimized via stochastic gradient descent in this model. Next, they created a neural network using TensorFlow and used three layers. They used one-hot encoding, as the data are categorical and a SoftMax output layer. Moreover, they used another neural network with two hidden layers, the model is stated like LINEAR > RELU > LINEAR > RELU > LINEAR > SOFTMAX. They also used Adam optimizer to compute cost. Furthermore, researchers used ten distinct Bangla typefaces to train their model. They also created an UI to demonstrate their research.

In [2], researchers proposed a technique for recognition of cursive Bangla characters using convolutional neural networks with a recurrent neural network. Here, CNN is similar to a transformation function in that it takes input image sequence frames and outputs a feature vector. A stochastic gradient descent approach is used to train the CNN. Next, the output vector is fed into the bidirectional recurrent neural network that uses LSTM as a hidden layer. A Connectionist Temporal Classification layer is located at the top of the recurrent net, at the end of the model to label sequences by obtaining a character recognition probability.

In paper[7], researchers proposed a system based on Tesseract OCR Engine with the user interface developed in the Java Graphical User Interface platform using NetBeans IDE and the codebase is formed as a zip package. A software based on JRE 'jTessBoxEditor' is used for training Tesseract through which the text files have been converted to images and the training information is stored in a generated file with extension 'tr'. The set of possible characters is known with the help of a unicharset file and the font is in UTF-8 format. The character shape features can be clustered using the shape clustering, mf training programs that are part of the OCR engine. The algorithm of Tesseract uses search and fetch policy for generating output. If a character is matched, it gets the coordinates of the character in that image, extracts the characters and outputs the file. If there is no matching character, it extracts the character that is closely related to the character of the input

file but produces a mismatched result for a large character set.

In[6], a complete OCR system for printed Bangla text has been implemented with efficient ways involving line and word detection, zoning, character separations and character recognition. The skew determination algorithm is based on the existence of matra, which begins with finding all the connected components using DFS considering only the components having width greater than the average. The OCR system identifies and separates different lines of text from the image file. The next steps are finding the span of each word and character separation. Thirdly, a straightforward deletion of the matra is done which splits the character into sub segments. After the deletion of the topline and bottom line, it comes to separating each character also known as feature; the boundary of pixels within which are to be scaled is marked. After that, thinning is conducted that enhances the uniformity of character shaping a bit further. It has 3 processes- image cracking, darkening and zoom-in. Now, the single pixelated characters need to be scaled to uniform shape before being fed to the neural network. Finally, the researchers implemented a multi-layered feed-forward back propagation neural network. After training character sets of 3 fonts- Sutonny, Sulekha and Madhumati; the model is tuned to suit the edge cases. The result is an accuracy of over 90% which is acceptable with a prospect of further improvement.

In [16] this paper the authors have used shape decomposition method, chain code drawing and multi-layer perceptron (MLP) for recognizing compound Bangla characters. 128 compound characters using 41 classes were recognized. Instead of recognizing complex shapes, they recognized two basic characters from the compound character, which improves performance drastically as the number of classes reduces. The workflow is preprocessing > segmentation > group formation > shape decomposition > feature extraction > classification. Preprocessing is done using binarization, granular noise cleaning, headline truncation (to remove elongated headline strokes) and partial thinning. Then one of the most difficult tasks, segmentation takes place. The line, which connects two consonants, is detected for segmentation. Based on structural shapes, pattern, segmentation and reference line (Matra) compound characters, the dataset is divided into five groups. Depending on the vertical and horizontal cuts are performed, to get better proper decomposition.

Same compound character can be written in two ways mostly, they also took that under consideration and performed the operation accordingly. Then comes feature extraction, in this part pictures are normalized (49x49 pixels) and chain code histogram is generated. Then the input data is given to the MLP feed forward network. It uses backward-propagation for minimizing the errors. The input layer has 392 nodes and the output layer has 41 nodes. Parameter tuning takes place to decide the average stroke width of Bangla characters, as most of the strokes are vertical. They have found the average stroke width is 5. They used three models of which MLP achieved the best overall result of 88.74% accuracy.

In[3], they used modified ResNet-18 architecture for Bangla Handwritten language detection. They used ResNet as Bangla language has spiral and horizontal structure due to its cursive nature. The network architecture uses: Convolutional neural network layer, Dropout layer, max pool layer, batch normalization and Adam optimizer. Bangla Lekha Isolated and CMATER db3 datasets are used in this research paper. Data preprocessing in this paper uses noise reduction, median filter, edge thickening and resizing to square shape. To make the used datasets diverse it uses image distortion method. By experiments, it was found that input image size 112x112, dropout rate 0.2 and using Adam Optimize,r ResNet architecture could achieve 95.1% accuracy. From the confusion matrix we can see the model often confuses among the characters (both isolate and compound) and classifies them as wrong classes, it is due to the complex nature of Bangla Language structure.

Paper[13] used pre-trained OCR models such as AlexNet and GoogLeNet to recognize Urdu characters through transfer learning. They used AlexNet and GoogLeNet to classify 1000 labels and used Data Augmentation and Dropout techniques to prevent overfitting. It uses SoftMax as an output layer. From the results, it can be said AlexNet works much faster compared to GoogLeNet because it spends more time in the inception module looking for features. For offline character recognition, AlexNet is a better option by a long shot. They used mobile phones combined with Matlab to identify offline characters. The accuracy achieved by AlexNet and GoogLeNet are 96.3% and 94.7% respectively.

LSTM [20] is a special kind of RNN which can maintain its cell state by storing useful data. As time proceeds; it also updates useful data and deletes irrelevant data. Therefore it makes LSTM better suited for sequential data analysis and determining next word. There are three gates in an LSTM [Fig 2.1] architecture; forget gate, input gate and an output gate. These gates have been used to store, update, forget and output data. LSTM prevents the network from becoming biased or overloaded.

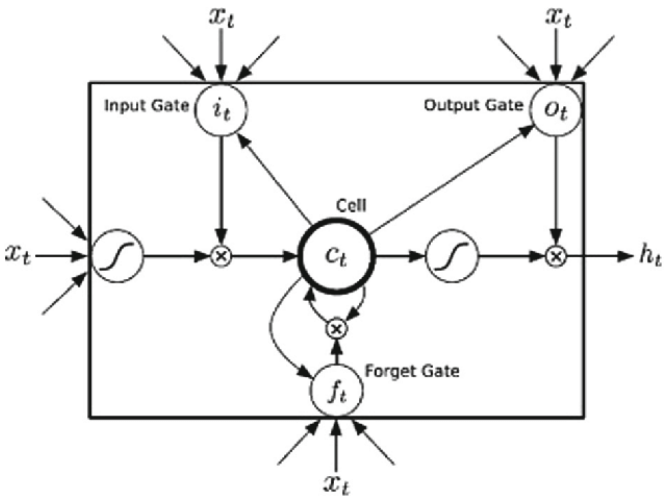


Figure 2.1: LSTM architecture

The softmax [5] layer predicts the sequence of text. It works as an output vector which normalizes the previous values to a probability between 0 and 1. Then the model makes a prediction depending on the SoftMax value. In[15] , the authors proposed a complete guide to build a Bangla OCR. The authors divided the steps into three categories; pre-processing, post-processing and classi-

fication. In the processing part, feature extraction divides the characters into geometric shapes and extracts the most prominent features about the characters. At last, when the data is affected with noise, Artificial Neural Network works better.

In [1], the authors have described an approach to recognize handwritten Bangla characters using a multilayer Convolutional Neural Network. The convolution blocks inside of the CNN architecture extracts the main features of an input image and reduces its dimension. Using a 3x3 kernel in the convolution layers is mostly popular. At first, the data is fed to the first convolution layer followed by a MaxPooling layer. A convolution block is basically a convolution layer followed by a MaxPool layer. After several convolution layers, they fed the data into an Inception Module with a fully connected CNN stacked parallelly and sequentially.

In [22], it is shown that the residual inception models are relatively more efficient in terms of computation power needed. It uses the inception blocks and a filter-expansion layer which helps scaling up the dimensionality to match the depth of the input. The Inception-ResNet-v1 has almost the same computational cost of Inception-v3 and the Inception-ResNet-v2 has the computation cost Inception-v4. It compares top-1 Error and top-5 error percentage of different Inception versions with similar cost Inception-Res-Net versions. And from the results it can be said that the Inception-Res-Net gives slightly better results.

Moreover, choice of datasets play an vital role in the practical life performance of the model. In [15], the dataset was collected using web scraping from “Prothom Alo” newspaper. As a result the dataset is somewhat biased, because newspapers are written in a certain style which does not necessarily follow day-to-day used communication semantics. Dataset should be diverse and distributed among different walks of life so that models can perform well in various situations.

Chapter 3

Methodology

Optical Character Recognition (OCR) can be divided into two phases; feature extraction and classification. In our research, we are using Inception-v3, ResNet-50_v2, and VGG16 for comparison purposes to test with our custom architecture model. Bangla character recognition is complex compared to other languages. Unlike English, as it has a cursive nature plus complex shapes of compound characters. At present, there are various techniques that are used for Bangla OCR where some of those did not get the optimum level of accuracy and could not satisfy different metrics criteria. Therefore, our desire is to analogize between the most prominent Deep Neural Network algorithms and perceive which one performs better while recognizing Bangla characters and also satisfies different metrics criteria. Our working methodology consists of the following parts:

Step 1: Data Collection
Step 2: Data Pre-Processing
Step 3: Data labeling
Step 4: Data Splitting
Step 5: Proposed Inception V3, ResNet-50 V2, and VGG-16 models.
Step 6: Illustrating the Performance of the Proposed Model
Step 7: Comparison between the different deep neural network architecture

3.1 Architecture

3.1.1 Our Proposed Model (KDANet)

For building our custom model, we took inspiration from multiple state-of-the-art models Inception_V3[22], Res-Net50_V2 [8], and Inception-ResNet[21]. For OCR we require a high-level feature deduction in addition to low and mid-level feature extraction. Deeper network reinforces the feature extraction for the multi-level classification process [24]. There are a few caveats to using Inception or ResNet alone for Bangla OCR:

1. **Overfitting Problem:** While training the Inception-v3 [22] network, we observed fluctuated differences continuously in validation loss and training loss which indicates overfitting. This problem was solved using early stopping and Regularization techniques. Moreover, ResNet50-v2 loss curves for both CMATERdb and BanglaLekha Isolated[4] show that even though training loss is decreasing in every epoch, validation loss keeps spiking after some epochs which proves that the ResNet50-v2 model is not performing well in our chosen datasets.

2. **Computational Cost:** Using ResNet for our dataset, each step took longer than the Inception model to train. This translates to increased computational cost. Moreover, among the three models we trained, VGG-16 was the most computationally expensive.

Inception ResNet[21] architecture solves these two crucial problems. By introducing factorization and skip-connection techniques, a fusion of ResNet and Inception Network happened. Factorization decreases the computational cost and a skip connection solves the vanishing gradient descent

problem which occurs when a network is profoundly deep. Due to computational complexity and hardware unavailability, it is not possible to implement such a Network. This paper proposes a model KDANet, inspired by Inception-ResNet. We have downscaled the architecture by eliminating duplicate layers and adding dropout layers. Moreover, we have specifically tuned hyperparameters for our OCR system.

3.1.2 Model Architecture

The proposed model takes a 224x224 pixel image as input and output the probability of 50 fundamental characters of Bangla language in the final fully connected layer. The root block is the initial layer, followed by Inception Resnet Block A, Reduction Block-A, Inception Resnet Block B and Reduction Block B, Inception Resnet Block C, a global average pooling layer, the fully connected layer with dropout, and lastly the softmax dense layer making a total of 229 layers.

The Root block[figure 3.2] is basically consecutive parallel convolutional layers, concatenated in 3 intervals. A batch normalizing layer and a ReLU activation function follow each Convolution layer. The output of the Root Block is then passed on to the Inception-ResNet blocks. Finally to implement dropout, a few dense layers were introduced which were then finally inputted into the softmax layer.

Reduction Block is the same as other convolution layers, except being a stride of 2x2 which increases the movement of the kernel and implies dimensionality reduction. It is safe to say that this is the reason why it is called the reduction block [21].
The Inception ResNet Blocks, Reduction Blocks and overall architecture are shown as follows. [3.1,3.2,3.3,3.4,3.5,3.6,3.7]

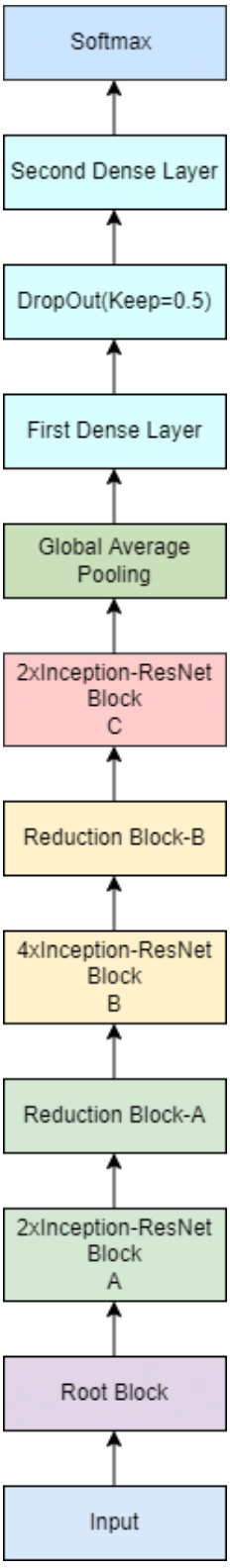


Figure 3.1: KDANet architecture (Proposed Model)

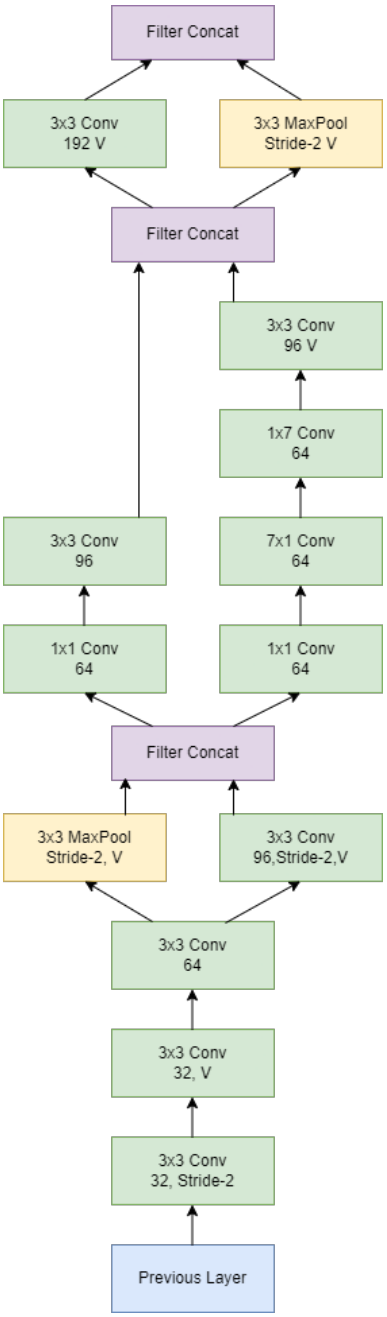


Figure 3.2: Root Block

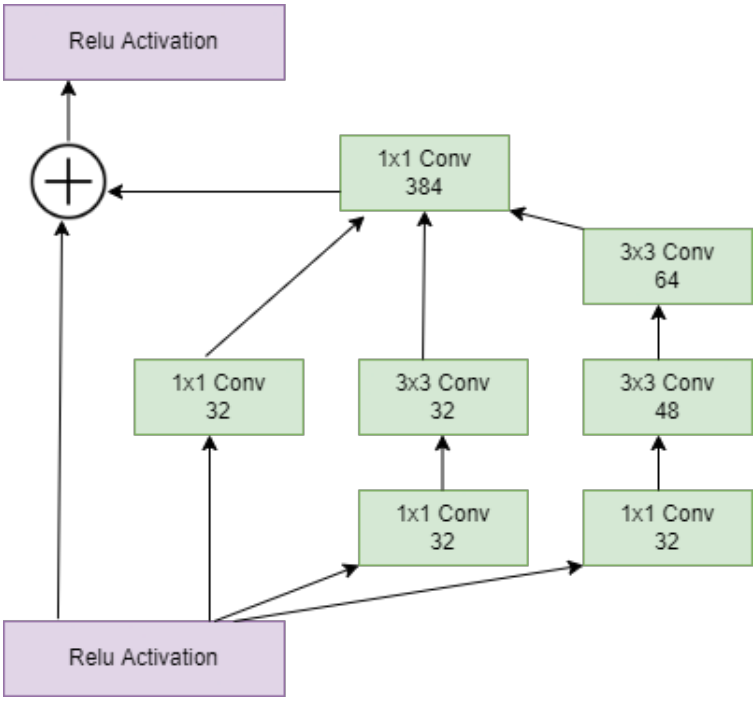


Figure 3.3: Inception ResNet Block-A

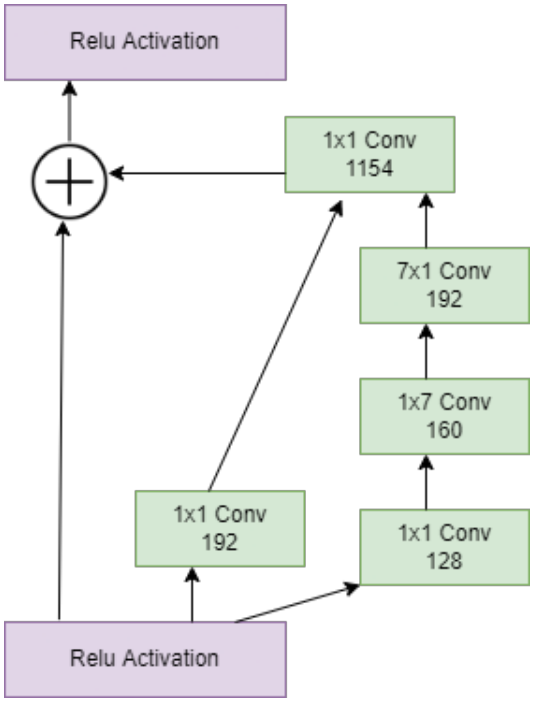


Figure 3.4: Inception ResNet Block-B

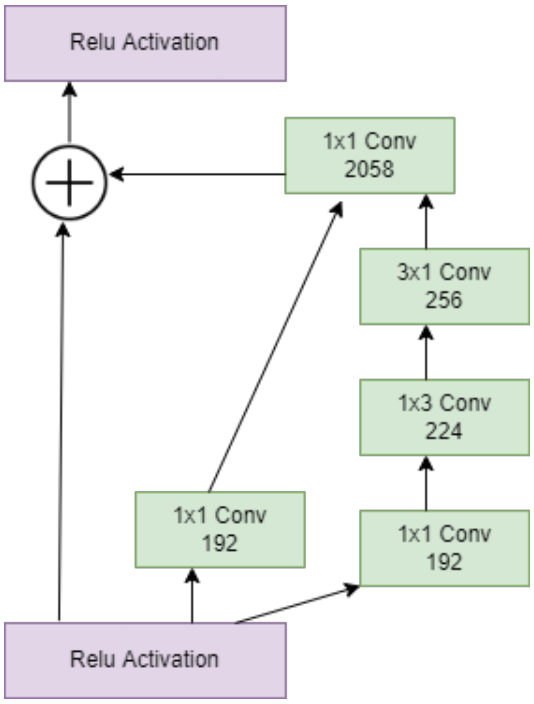


Figure 3.5: Inception ResNet Block-C

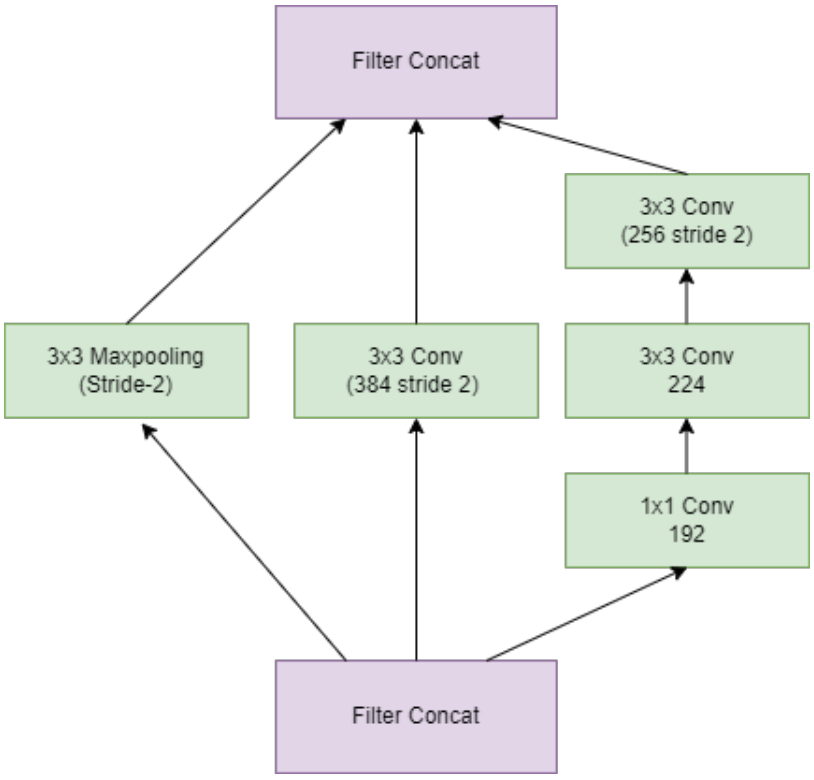


Figure 3.6: Reduction Block-A

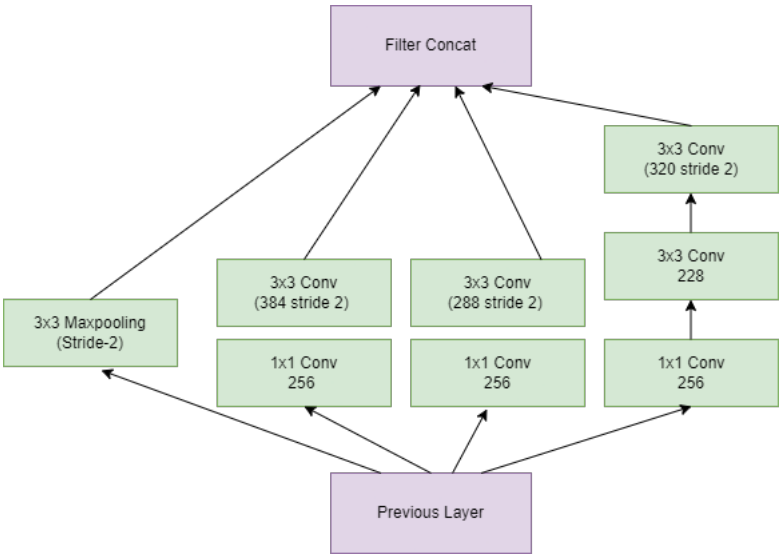
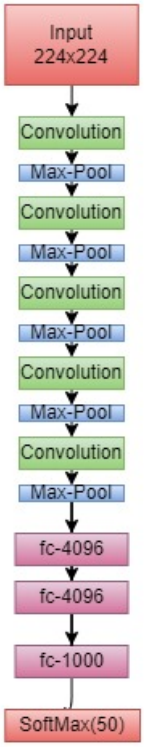


Figure 3.7: Reduction Block-B

3.1.3 VGG16

CNN works best for feature extraction for image data. To further improve it, a deeper convolutional network was introduced in [18]. VGG-16 is a very deep state-of-the-art neural network architecture having 16 convolution layers and 3 fully connected layers. The outer layers of a deeper network learns normal features (lines, curves, etc.) and the deeper layers of the network learn abstract features like shapes. Moreover, 3x3 and 1x1 kernel sizes were used over 7x7 which was previously used. The reason is that it helps to reduce parameters and linearity in the model. Instead of using a 7x7 kernel, three 3x3 kernels were used along with a ReLU activation function. Usage of 3 ReLU units rather than one alongside a 7x7 kernel helps to introduce non-linearity in the model. 1x1 and 2x2 strides are used with convolution Layer and max pool layers with 5x5 kernel respectively. Techniques used in VGG-16 are MaxPooling, Dropout, mini-batch training, and ReLU.

As OCR requires a very deep understanding of character shapes, it is essential to use a network that can extract regular and abstract features, especially for a language like Bangla which has very closely matched characters. So, it is important to understand abstract features as well.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3.8: VGG-16 architecture

3.1.4 Resnet50-v2

In the paper [8]"Deep Residual Learning for Image Recognition" (2015), the researchers paved the way for Res Net with so many different variants. The problem which arises when training deep neural networks is the vanishing gradient, which was a matter of concern for deep learning practitioners. To tackle this, lots of techniques started to spread across researchers mostly by sacrificing performance and no solution was without its side-effects.

The resolution came in the form of Residual networks which are basically combinations of residual blocks where a skip connection is at the heart of all. The technique is to skip some layers with a direct connection which ultimately changes the output with necessary dimension adjustments. Here an identity function also comes into play to stop the later layers to perform worse. The residual networks too provided the practitioners to train much deeper neural networks without the concern of worsening performance unlike other architectures.

The variant we have selected for our OCR system is ResNet-50. Initially, ResNet-34 was developed inspired from the highly efficient VGG-19 architecture, but had one deep flaw- very long training time. This has been fixed with the 50 layered variant by making use of a stack of 3 layers as a bottleneck block in place of the previously used 2-layer block. Upon further research, ResNet-50 has demonstrated itself as a prominent neural network architecture in the fields of computer vision which prompted us to utilize this for our OCR system. However, we did need to imply tuning of hyper-parameters and parameters heavily.

Layer name	Output Size	50-layers breakdown
conv1	112x112	7x7, 64, stride 2
conv2_x	56x56	3x3, max pool, stride 2
conv3_x	28x28	[1x1 , 64 3x3 , 64 1x1 , 256] X 3
conv4_x	14x14	[1x1 , 128 3x3 , 128 1x1 , 512] X 4
conv5_x	7x7	[1x1 , 256 3x3 , 256 1x1 , 1024] X 3
	1x1	Average pool, 1000-d fc, softmax
FLOPs		3.8×10^9

Figure 3.9: RESNet50-v2 architecture

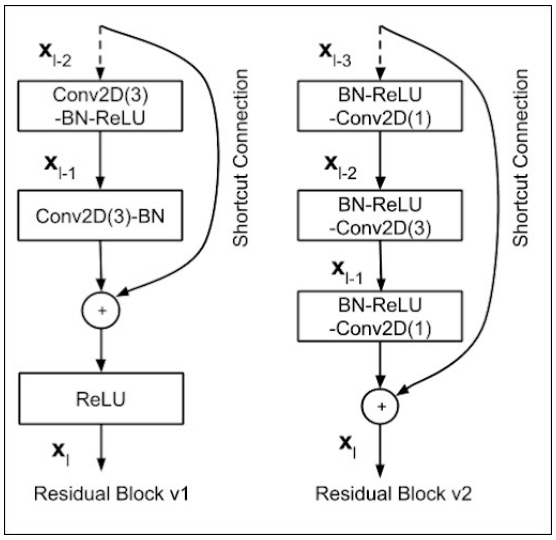


Figure 3.10: Shortcut Connection ResNet

3.1.5 Inception-v3

[22]Deep convolutional neural networks have been chosen for different tasks over the past years as it has shown significant improvements in a variety of benchmarks. However, with deeper neural networks the computation cost of the model becomes expensive. Moreover, one of the challenging issues in Convolution Neural Network is selecting the correct kernel size. To spread information globally, a larger kernel is recommended, whereas for information that is dispersed more locally, a smaller kernel is recommended. To overcome these issues, the Inception module was introduced. The inception module uses three distinct filter sizes (1x1, 3x3, 5x5) to conduct convolution on an input which solves the issues of choosing the right kernel size. Additionally, max pooling is applied. Concatenated output are forwarded to the subsequent inception module. Furthermore, to make the computation less expensive, a 1x1 convolution layer was introduced before the large 3x3 and 5x5 convolutions.

To reduce the computational cost even more, inception-v2 and inception-v3 were introduced. Here the large convolution layers are factored into small convolutions. For instance, 5x5 convolutions were factored into two 3x3 convolutions which reduced the computation costs as 5x5 convolutions are 2.78 times more costly than 3x3 convolutions. Moreover, authors factorize the NxN convolution into 1xN and Nx1 convolutions and this method proved to be 33% cheaper than NxN convolutions. Moreover, in inception-v3 RMSprop optimizer and a regularization component was introduced to prevent overfitting. For these benefits over other algorithms, the Inception-V3 algorithm has been used for testing in our research paper. Techniques used in Inception-V3 are max pooling,dropout,batch normalization and ReLU activation function.

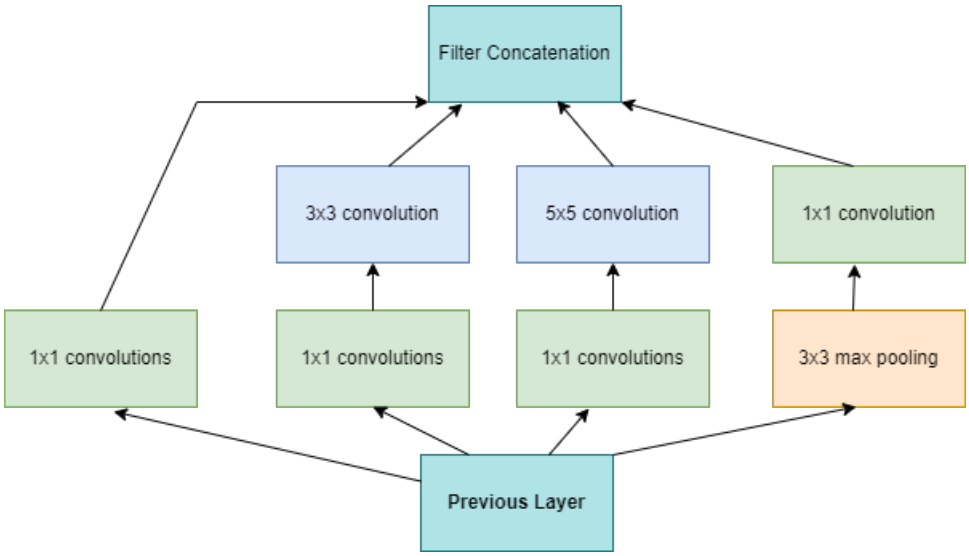


Figure 3.11: Inception Kernel[21]

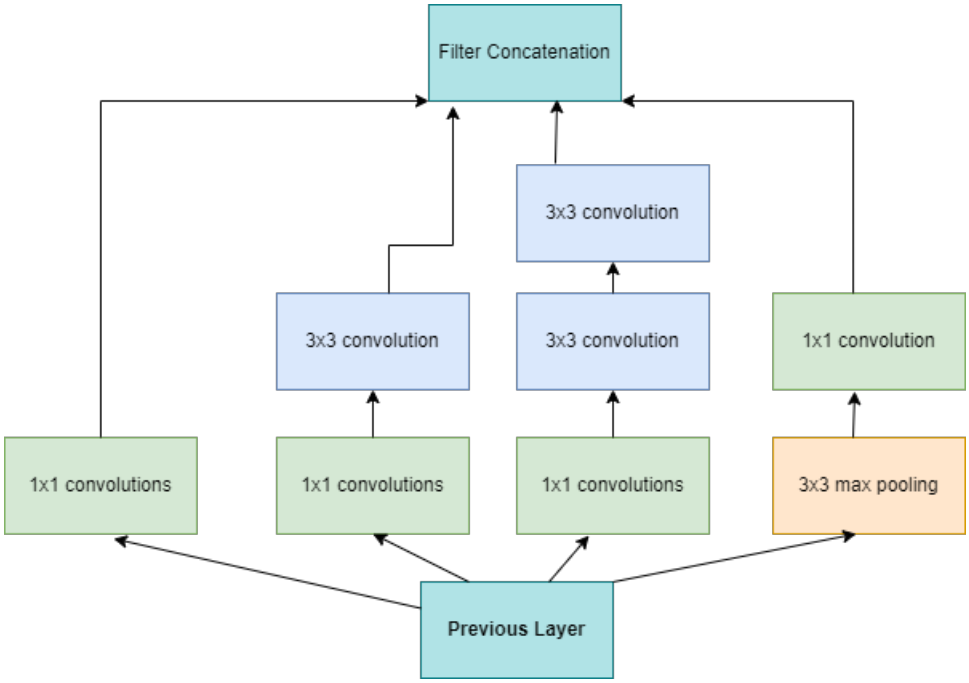


Figure 3.12: Inception Kernel Reworked[21]

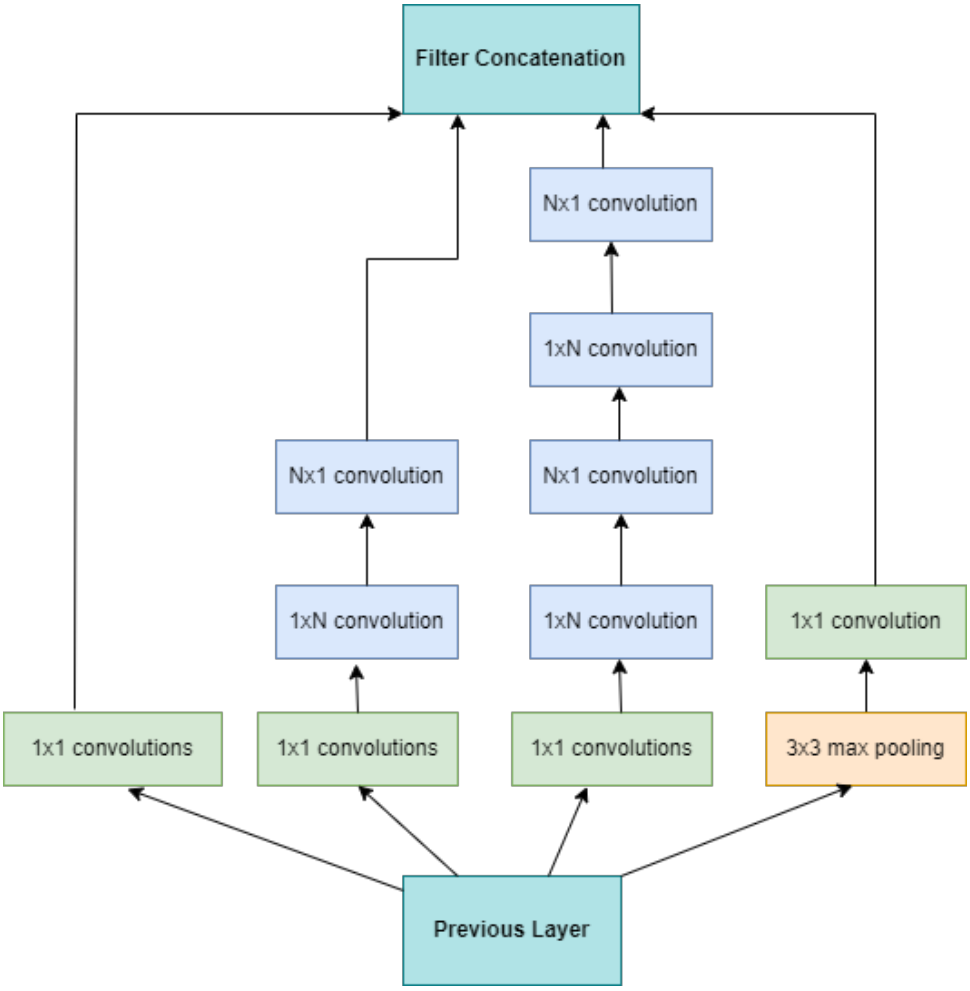


Figure 3.13: Factorized Kernel[21]

3.2 Tuning and Optimization Techniques

3.2.1 ReLU

[14]The Rectified Linear Activation function (ReLU) is a popular activation function for various kinds of deep neural network models. It provides non-linearity to the neurons. It is a non-linear activation function which works on a very simple logic. It always takes the maximum value between 0 and the output. Therefore, it outputs the input when the input is positive, otherwise it outputs 0. The ReLU function is a very useful tool to activate only the neurons that have a positive linear transformation value.

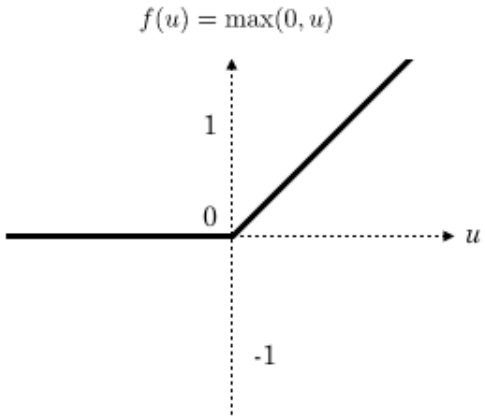


Figure 3.14: ReLU Activation Function

3.2.2 SoftMax

[5]Softmax functions are widely used as a classifier's output to express the probability distribution across n classes. Mostly it is positioned as the last layer of a multi-class deep learning model. This function calculates the probability of an input for each possible class. It can be mathematically described as-

$$softmax(Z)_i = \frac{exp(Z)}{\sum_j exp(Z)_i} \tag{3.1}$$

Here, Z is the input of the softmax function which is the value of neurons of the output layer of a multiclass mode. The exponential function adds non linearity. The normalization term $\sum$exp(Z) ensures that the sum of all the probabilities for an input is equal to 1. As our problem is a multi-class classification one, that is why to classify and predict in which class a character is in, softmax activation function has been used.

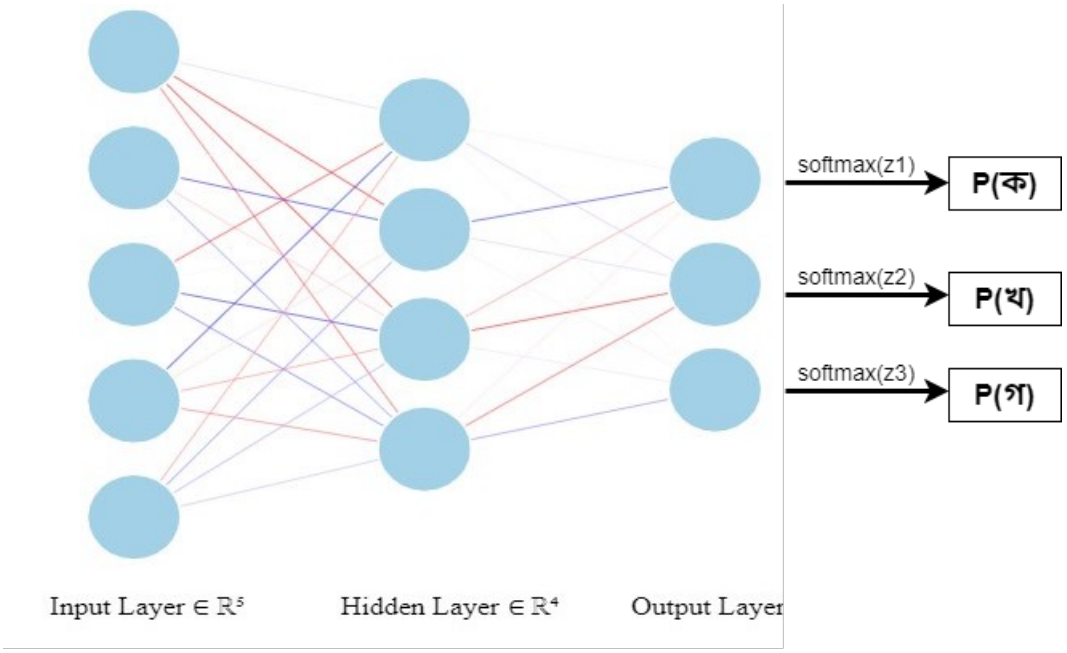


Figure 3.15: SoftMax

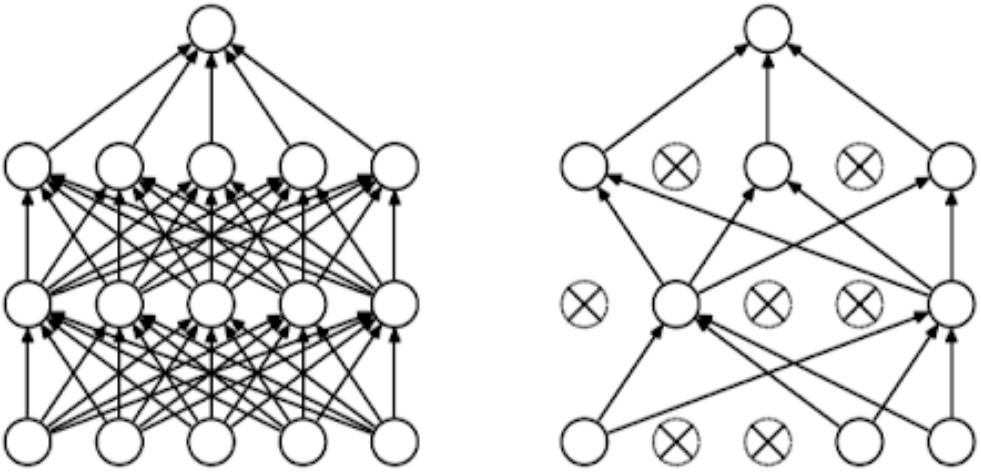
3.2.3 Dropout

[19]Dropout is a widely used regularization technique that averts the overfitting problem. As the architecture of neural networks gets deeper, it tends to learn more than the smaller ones. This results in an overfitting problem. [19] introduced this technique that disconnects the hidden units of a network while training. It is done randomly and as such, the network doesn't learn unnecessarily. As a result, it performs better in the unseen dataset and performs better. A vector of Bernoulli's random variable having the probability is multiplied by the output layer. Then depending on the value, the hidden units will be disconnected from the rest of the network.

$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^{l} + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

Figure 3.16: Dropout Equation

This novel technique can also be referred to as the thinning of the network. Dropout =0.5 is widely used in different models, it translates that 50% of the hidden units will be randomly disconnected. For OCR we are using deep architecture, deep models tend to learn more and overfit. For this reason, we are using the dropout technique to improve performance.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.17: Dropout

3.2.4 Adam Optimizer

[12]The prime optimization algorithm for training deep neural networks had been initially stochastic gradient descent. However, it had some significant drawbacks especially when dealing with Big data and then to further optimize this, the Adaptive Moment Estimation algorithm came into being. It's basically a combination of two prominent gradient descent methodologies, 'adaptive gradient' and 'root-mean-square propagation' by taking the best of both of these.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Figure 3.18: Adam Optimizer Equation

The special feature of the Adam algorithm is the simplistic configuration and the handling of sparse gradients on noisy data. Aside from that, intuitive as it is, it is also quite memory efficient and proven to be highly effective in practice. Here the parameters to focus here are alpha, beta1, beta2 and epsilon.

3.2.5 MaxPool

[23]Max Pooling downscales an image to only include the most prominent feature from the convolutional filter. Thus, the output would be a feature map with the features that are only essential from the preceding feature map.

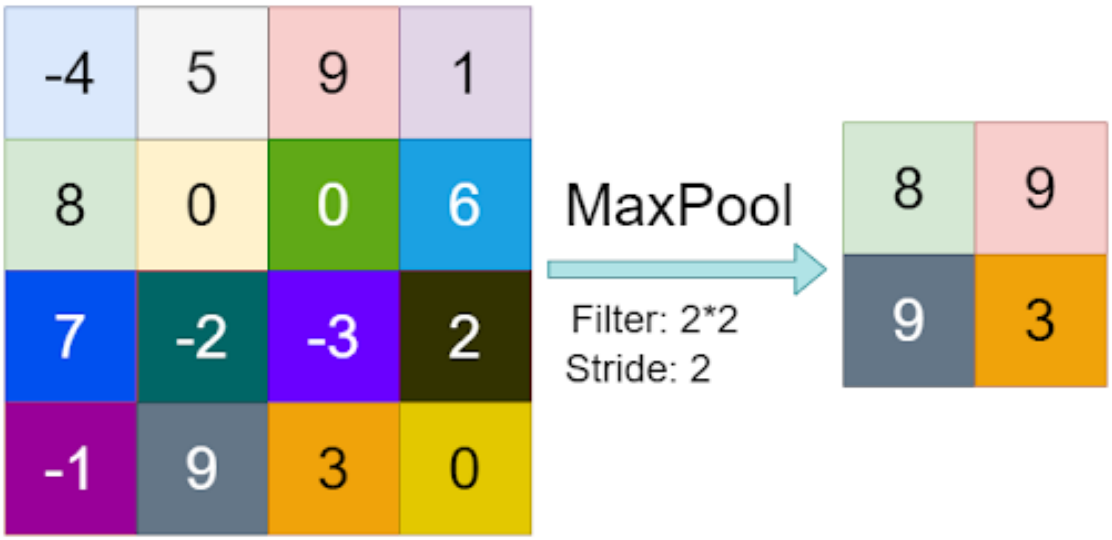


Figure 3.19: Max Pooling

3.2.6 Early Stopping

Our aim while training a deep neural network is to produce the best possible results. However, very large deep neural networks are apt to overfitting. Overfitting occurs when a model performs well in training data, but not in validation data or unknown data. [17]Early stopping is a strategy

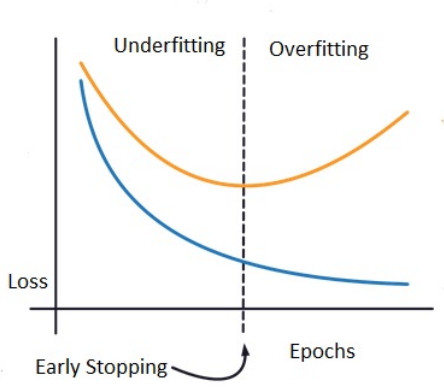


Figure 3.20: Early Stopping

for avoiding overfitting in which training the model is stopped after specific conditions are met. Validation loss was selected to end training early in this study. As validation loss began to increase substantially while training error was decreasing, early stopping was called.

Chapter 4

Dataset

4.1 Data Collection

When it comes to Optical character recognition for the Bangla language, there are not too many available options to begin with, but there are a few. CMATERdb 3.1.2 and Bangla Lekha Isolated are two of the most renowned datasets which contributed to the research regarding the Bangla language by providing clean datasets. There aren't many printed datasets available for Bangla characters. Most of the available datasets don't contain compound characters, which is another big problem. There are more than 150 compound characters [16]. For the sake of simplicity and to get an overview of performance metrics, we considered a dataset of isolated characters as it shortens the number of classifications.

CMATERdb is a repository for Bangla Pattern recognition-based works. It is an open-source database. We are explicitly using CMATERdb 3.1.2. It includes images of 11 vowels and 39 consonants which are handwritten characters with white and black texts on it (grayscale image). It includes a total of 15000 images of Bangla characters. 12000 of them belong to the training class and 3000 of them belong to the test class. There are 50 directories of 11 vowels and 39 consonants each of them contains 60 images. Every image is in .bmp format.

BanglaLekha-Isolated dataset [4] is a collection of handwritten Bangla characters and numerals. It contains 50 characters, 10 numerals, and 24 compound characters. There are a total of 1,66,105 images. For getting an overview of how the models will perform, we extracted 10 Bangla numerals and used them to train the models. Each of the 10 directories of numerals contains 1940 images. Every image has e black background and white font on them. We used an 8:2 ratio on this dataset for training and testing. The dimension of the images is not consistent. The dataset also includes information about the age and gender of the subjects from whom the handwriting samples were collected.

4.2 Data Pre Processing

To pre-process data, first, we need a quality check of the dataset. As the data is taken from different sources and taken in different shapes. Realistically it will contain missing values, duplicate values, and even inconsistent values. While reading the CMATERdb dataset, we found data inconsistency in the test data using trial and error method.

After finding the inconsistent data, we removed the bug from the dataset which was an extra hidden file. Initially, our dataset has a total of 15000 images with 12000 training and 3000 testing. However, the pixel dimensions of the images found from our dataset were not fixed, which could be an issue for maintaining balance in data. So, we rescaled our images to 224x224 for training all of the models.

We also used data-augmentation techniques to assign random height, random width and random rotation. Data augmentation helps to expand the diversity of the trainable data for the model training. It also increases the amount of data since it generates new data points from the ones already existing.

We also tried random zoom as another data augmentation technique, but since some of the Bengali characters are very similar (অ and আ, উ and ঊ); while cropping in, some features get lost; which results in the character being a complete different one. Therefore, we decided not to do random zoom.

Figure 4.1: BanglaLekha-Isolated data samples with corresponding printed characters

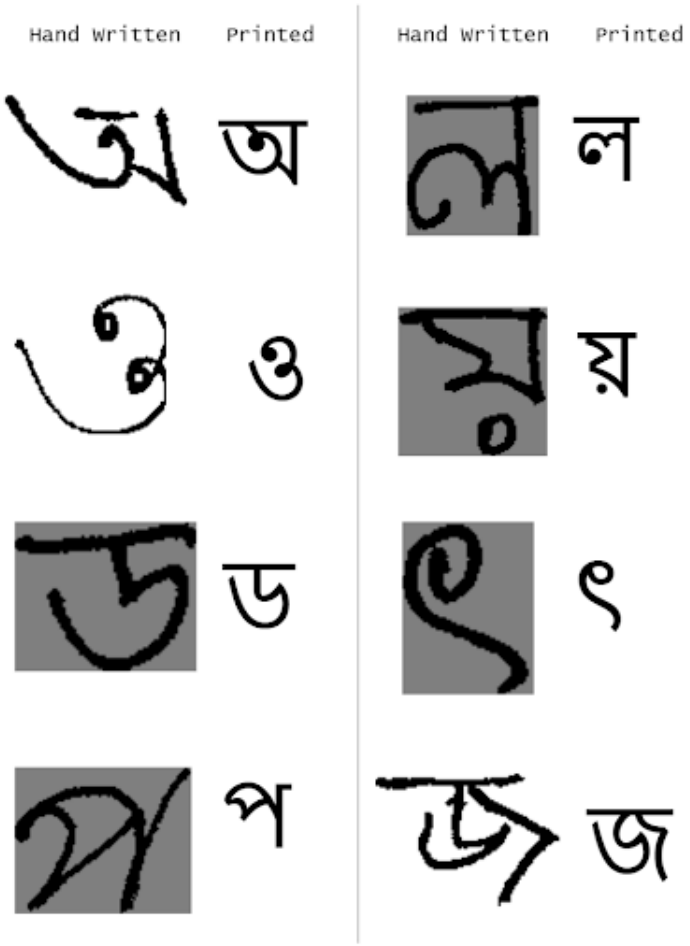


Figure 4.2: CMATERdb data samples with corresponding printed characters

4.3 Data Labeling

CMATERdb 3.1.2 dataset has 50 directories both in test and train directories. Directories are labeled from 172 to 221 each directory contains unique alphabets of the Bangla language. We used directories to label alphabets as their directory has a unique name. Each train directory contains 240 images and each test directory contains 60 images in this database.
BanglaLekha-Isolated[4] dataset has 84 directories in both train and test set labeled from 1 to 84. Each directory is labeled distinctly and contains images of a unique alphabet of Bengali language. To compare our model's evaluations with CMATERdb 3.1.2, 50 directories from BanglaLekha-Isolated[4] have been separated. Each train directory contains 1450 images and each test directory contains 250 images chosen for our research purpose.

Chapter 5

Implementation and Result Analysis

For model evaluation, we used 2 different datasets. BanglaLekha-Isolated and CMATERDB. All the models were trained with these two datasets and we compared the results.

5.1 Implementation Setup

This research used python programming language as the python has proven to be one of the best for machine learning and data-science research due to its simplicity, as well as it has a large range of accessible libraries and frameworks. In our research, the tensorflow framework is being used to implement the Inception-V3, ResNet50-V2 and VGG-16 models. Tensorflow library is quite popular among the deep learning practitioners. Tensorflow also has GPU compatibility to run models faster than CPU. Moreover, for plotting charts we have used Matplotlib library and for performing different tasks we used NumPy, Scipy and Sklearn libraries in our research.

We have used two computers with configuration i5 8400 GTX 1660 Ti and i7 7700HQ GTX 1050 ti configuration to train our model.

5.2 Peformance Analysis

In our research, for training and testing stages, we displayed loss and accuracy comparison for each epoch. It allows us to see how our model performs in each epoch for training and testing. It depicts if the model is overfitting or underfitting.

In the following figures, the accuracy and loss curves have been shown for both CMATERdb and BanglaLekha Isolated dataset.

First of all, for the CMATERdb dataset, the loss and accuracy curve for all the models have been shown in the following diagrams. For ResNet50-v2, the model was overfitting so we used early stopping after 10 epochs. The graph visualizes the gap between the training loss curve and validation loss curve which indicates that the model is overfitting at the start but after some time, the model was performing well. However, after 10 epochs it began to overfit again which was handled using early stopping. The accuracy on training data for this model is 98.43% and the validation accuracy is 97.10%. Moreover, for the VGG-16 model the training accuracy is 98% and validation accuracy is 96.2%. Here, for both the training and testing phases, the gap between the lines is also modest for both training and loss curve that indicates that this model is performing well. Finally, in the Inception-v3 model the training accuracy is 98.31% and validation accuracy is 97.93%. The accuracy graph visualizes that the validation loss and the train loss is close to each other after a few epochs and the accuracy graph shows a good performance on the dataset.

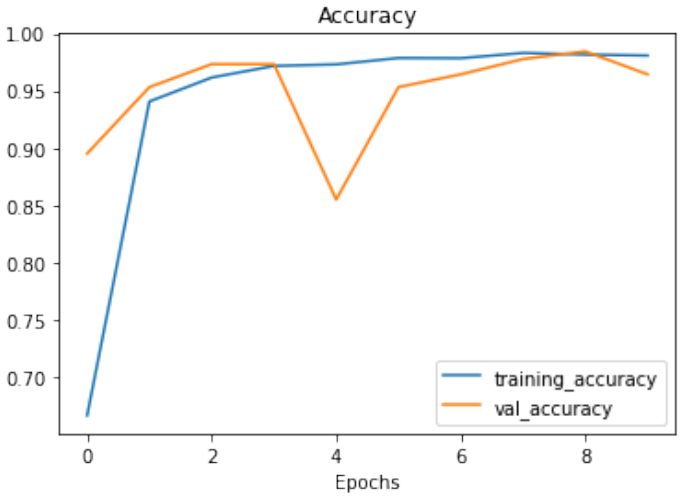


Figure 5.1: ResNet50 v2 accuracy (On CMATERdb dataset)

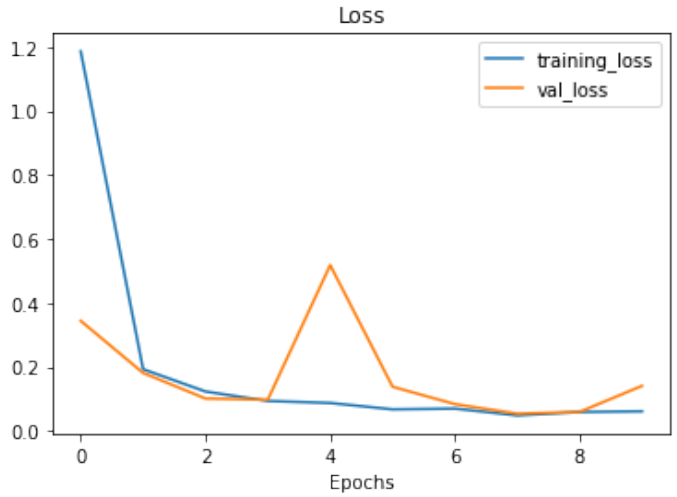


Figure 5.2: ResNet50 v2 loss (On CMATERdb dataset)

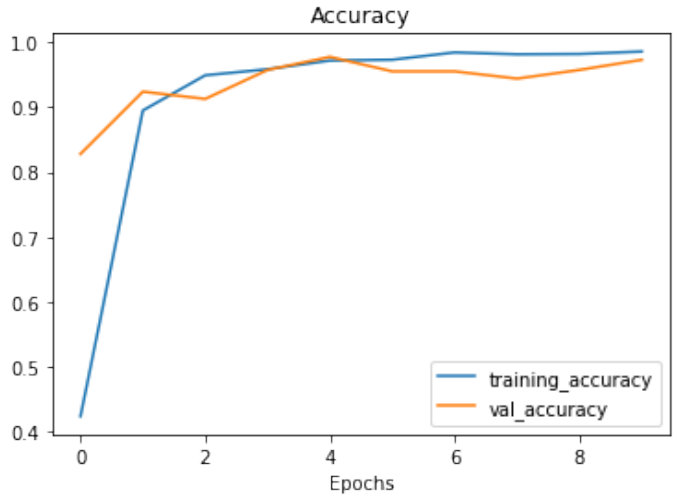


Figure 5.3: VGG-16 accuracy (On CMATERdb dataset)

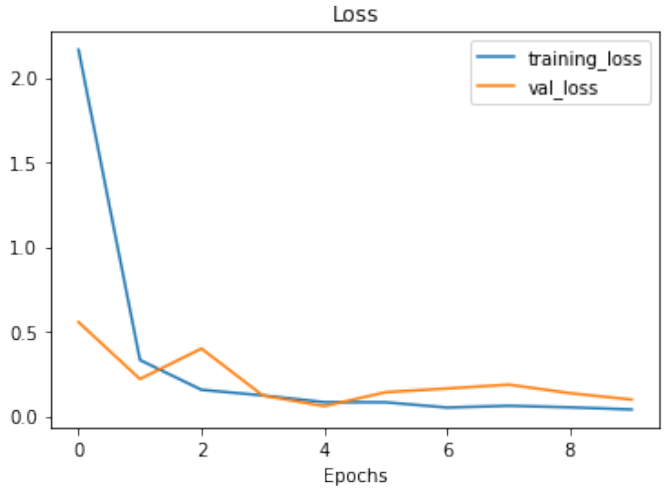


Figure 5.4: VGG-16 loss (On CMATERdb dataset)

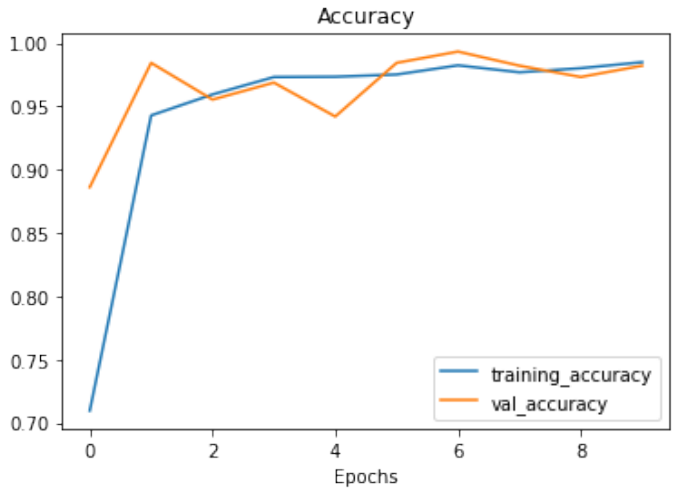


Figure 5.5: Inception v3 accuracy (On CMATERdb dataset)

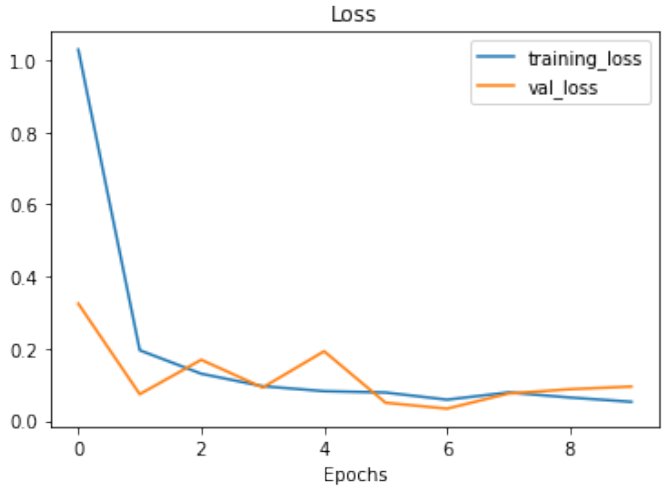


Figure 5.6: Inception v3 loss (On CMATERdb dataset)

The loss and accuracy curve for BanglaLekha Isolated on all the models have been shown in the following diagrams. These models have been trained with a batch size of 256 with a learning rate of .0001. BanglaLekha isolated models were trained for 20 epochs. After 20 epochs the Inception-v3 model showed 98.07% train accuracy and 97.68% validation accuracy. This model used early training to stop its training after 20 epochs as it was going to overfit. From the loss curves, we can see that the validation loss curve started to go up while the training loss was going down. Moreover, the ResNet50-v2 model had 97.86% train accuracy and 96.7% validation accuracy. ResNet50-v2 encountered an overfitting issue as the loss graph shows. As already mentioned, we used early stopping to stop the model to learn too much on training data. Finally, the VGG-16 model has 98.80% train accuracy and 97.52% validation accuracy. However, VGG-16 performed well in the BanglaLekha Isolated dataset as the loss curve shows.

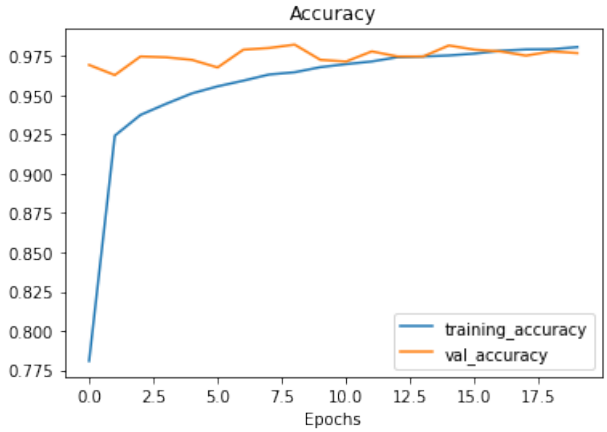


Figure 5.7: Inception v3 accuracy (On BanglaLekha Isolated dataset)

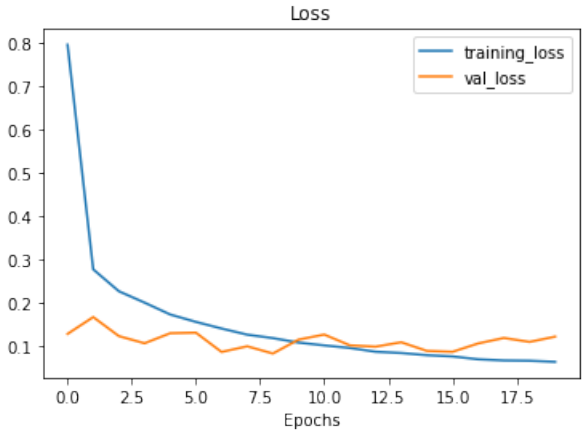


Figure 5.8: Inception v3 loss (On BanglaLekha Isolated dataset)

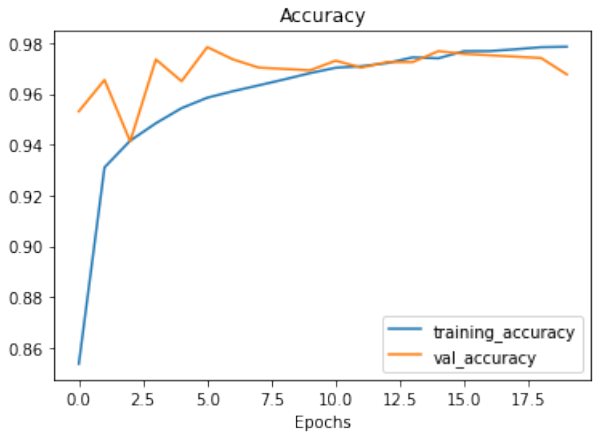


Figure 5.9: ResNet50 v2 accuracy (On BanglaLekha Isolated dataset)

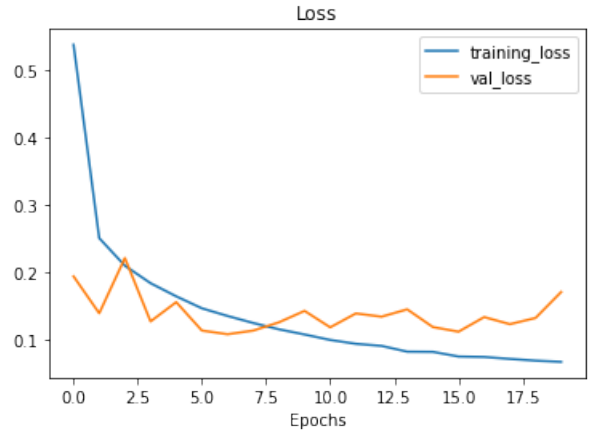


Figure 5.10: ResNet50 v2 loss (On BanglaLekha Isolated dataset)

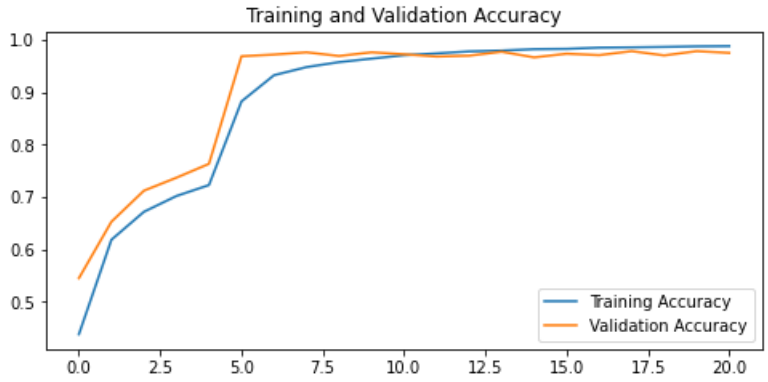


Figure 5.11: VGG-16 accuracy (On BanglaLekha Isolated dataset)

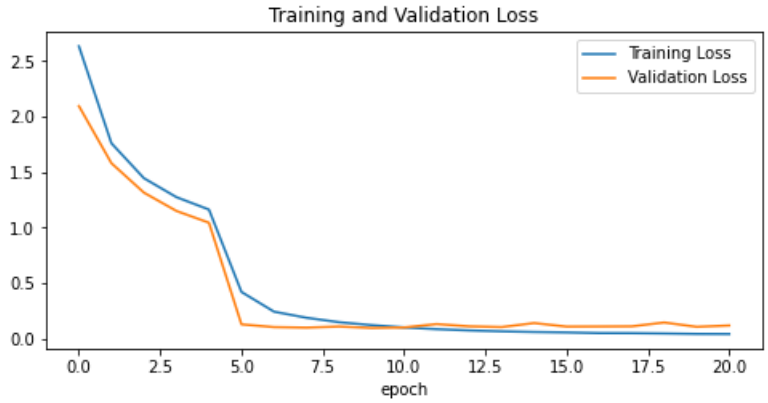


Figure 5.12: VGG-16 loss (On BanglaLekha Isolated dataset)

The following figures illustrate the accuracy loss curves of our proposed custom model KDANet. In the case of BanglaLekha Isolated dataset, KDANet had a training accuracy of 95% and validation accuracy of 96.71%. Moreover, from the loss curves, we can see that the curves are close to each other which explains that our model is not overfitting. Regularization and dropout techniques have been used so that we can avoid overfitting issues. On the CMATERdb dataset, the training accuracy is 95.48% and a validation accuracy of 95.76%. Also, the loss curves also show that the model is converging to the optimal point. However, to compare with other models, the training was stopped after 10 epochs. Furthermore, to see how our model performs if we train our model for more time, we trained our model for 40 epochs on BanglaLekha Isolated dataset. After 40 epochs, the training accuracy increased to 97.94% and 97.79% validation accuracy. The loss curve shows how well the model is performing on our KDANet model.

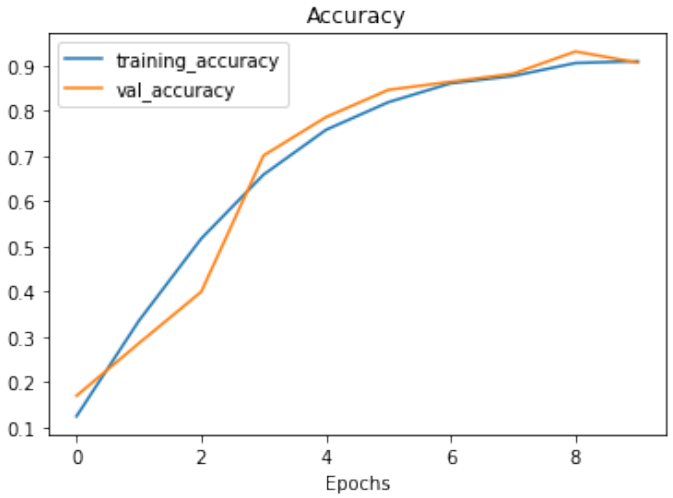


Figure 5.13: KDANet(proposed model) accuracy (On CMATERdb Isolated dataset)

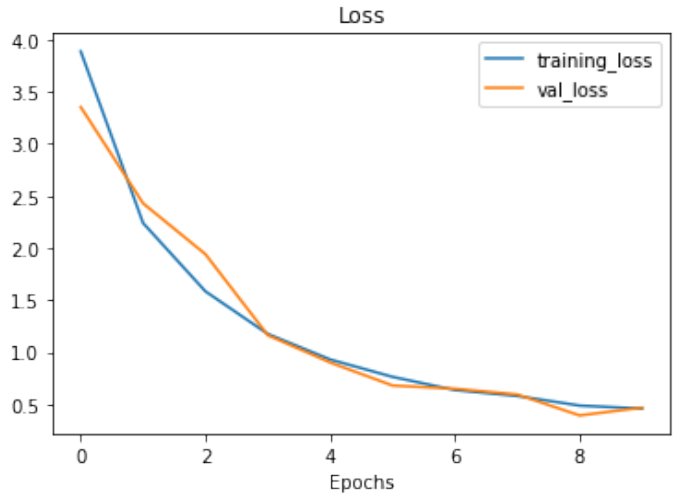


Figure 5.14: KDANet(proposed model) loss (On CMATERdb Isolated dataset)

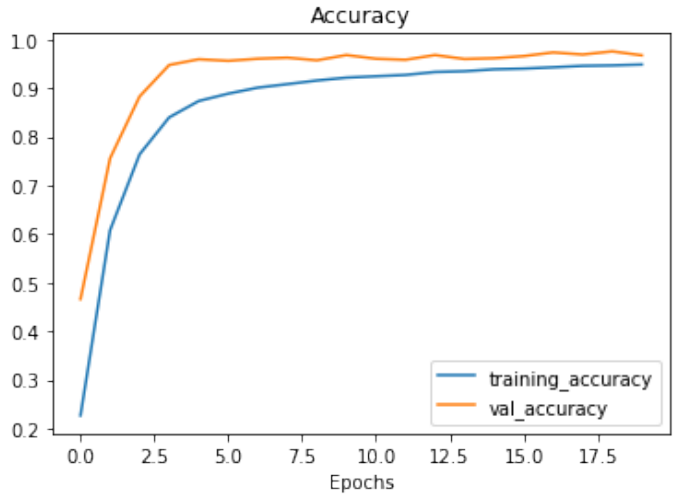


Figure 5.15: KDANet(proposed model) accuracy (On BanglaLekha Isolated dataset)

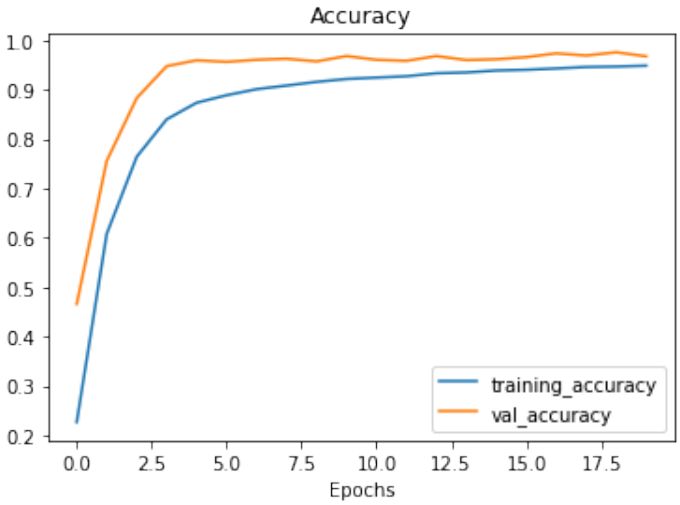


Figure 5.16: KDANet(proposed model) loss (On BanglaLekha Isolated dataset)

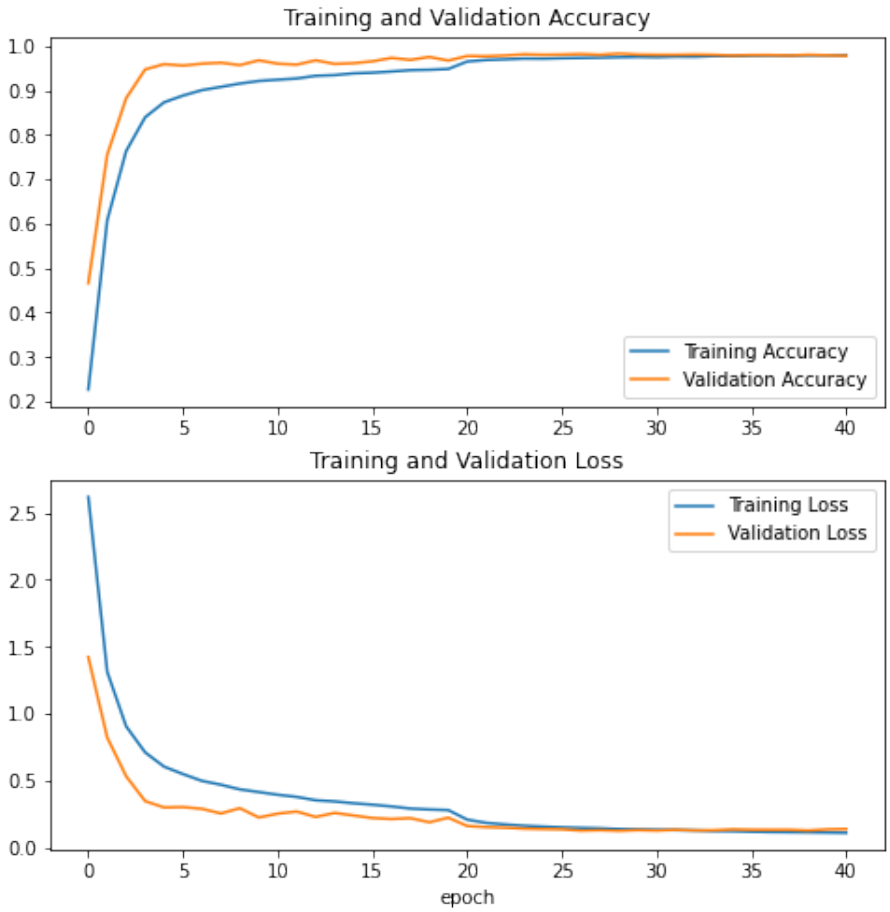


Figure 5.17: KDANet(proposed model) Accuracy and Loss Curves (After 40 epochs)

A confusion matrix is a means of analyzing a classification algorithm's performance. It can help to visualize what a classification model gets right and what errors are being done by the model. The following diagram shows a part of the confusion matrix of CMATERdb dataset. The matrix shows the amount of test data it got correct and which mispredictions it performed. For character 'ঘ', the matrix shows that the model identified 57 out of 60 test images correctly and the rest of the images were mispredicted as 'খ' . The reason behind the mispredicting can be resolved if we look at the dataset; some of the characters of class 'ঘ' looks exactly like like 'খ' as if you miss header stroke (Matra) from ‘ঘ’ it looks a lot like ‘খ’.We will discuss more about similar looking characters later on.

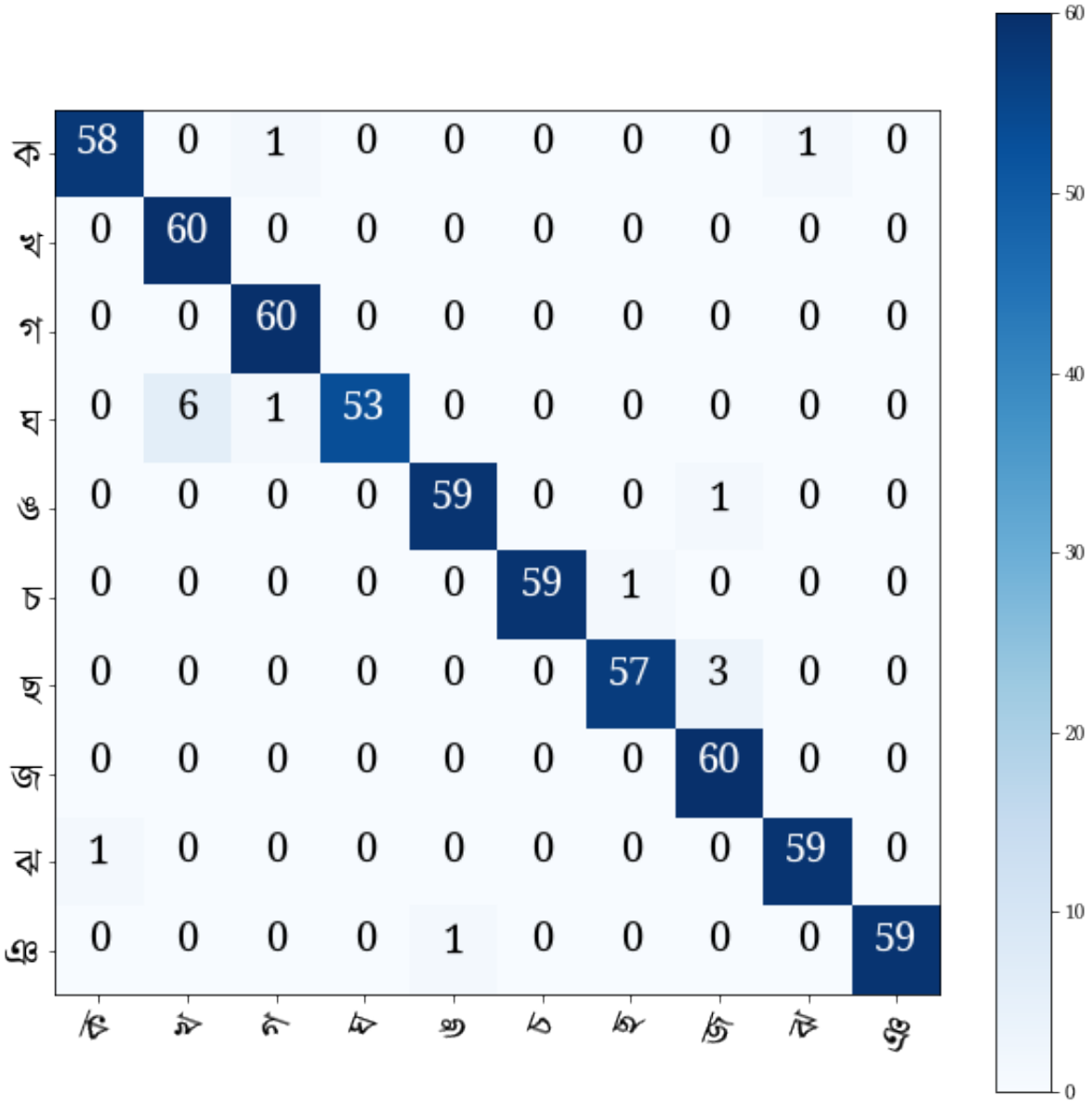


Figure 5.18: Figure: A portion of the Confusion Matrix(CMATERdb dataset)

Characters	Inception_V3	ResNet50_V2	VGG-16	Proposed Model
অ	0.99	0.98	0.97	0.99
আ	1	0.97	0.99	0.98
ই	0.99	0.99	0.99	0.99
ঈ	0.98	0.99	0.99	0.99
উ	0.96	0.96	0.96	0.95
ঊ	0.97	0.97	0.98	0.97
এ	0.93	0.92	0.92	0.94
ঐ	0.99	0.98	0.99	0.97
ও	0.98	0.98	0.96	0.98
ঔ	0.99	0.98	0.99	0.99
ঋ	0.99	0.98	0.99	1
ক	0.99	0.98	0.99	1
খ	0.98	0.98	0.99	0.96
গ	0.99	0.99	0.98	0.99
ঘ	0.99	1	0.99	0.99
ঙ	0.99	0.98	0.94	0.98
চ	0.96	0.94	0.95	0.95
ছ	0.97	0.95	0.97	0.96
জ	0.96	0.94	0.94	0.95
ঝ	0.99	0.96	0.98	0.98
ঞ	0.97	0.97	0.97	0.96
ট	0.99	0.98	0.99	0.99
ঠ	0.99	0.95	0.98	0.99
ড	0.97	0.97	0.94	0.96
ঢ	0.96	0.94	0.93	0.96
ণ	0.98	0.98	0.99	0.98
ত	0.99	1	0.98	0.99
থ	0.96	0.98	0.98	0.97
দ	0.95	0.93	0.94	0.95
ধ	0.95	0.94	0.92	0.96
ন	0.93	0.91	0.93	0.94
প	0.98	0.99	0.99	0.99
ফ	1	1	0.99	0.99
ব	0.99	0.96	0.98	0.99
ভ	0.98	0.98	0.99	0.98
ম	0.98	0.98	0.97	0.97
য	0.96	0.94	0.94	0.96
র	0.99	0.98	0.99	0.99
ল	0.99	1	1	1
শ	0.98	0.98	0.99	0.99
ষ	0.99	0.98	0.98	0.99
স	1	1	1	0.99
হ	1	1	1	0.99

Characters	Inception_V3	ResNet50_V2	VGG-16	Proposed Model
ড়	1	1	1	0.99
ঢ়	0.97	0.97	0.98	0.97
য়	1	0.94	0.99	0.99
ৎ	0.98	0.96	0.98	0.98
ং	0.99	0.99	0.99	0.96
ঃ	0.98	0.96	0.99	0.98
ঁ	0.98	0.98	0.98	0.98

Table 5.1: CMATERdb F1 Score

In the table [5.1, 5.2], we can see the **F1 scores** of Inception-v3, ResNet50-v2 and VGG-16 and our proposed model KDANet. In each row , we can see character and their corresponding F1 score in different models. The score 1 means the model was able to identify the character 100% accurately. Basically, it is the percentage of time the model was able to identify correctly. The scores gives us a detailed overview of the performance of models on individual characters.

Characters	Inception_V3	ResNet50_V2	VGG-16	Proposed Model
অ	0.99	0.98	0.97	0.99
আ	1	0.97	0.99	0.98
ই	0.99	0.99	0.99	0.99
ঈ	0.98	0.99	0.99	0.99
উ	0.96	0.96	0.96	0.95
ঊ	0.97	0.97	0.98	0.97
এ	0.93	0.92	0.92	0.94
ঐ	0.99	0.98	0.99	0.97
ও	0.98	0.98	0.96	0.98
ঔ	0.99	0.98	0.99	0.99
ঋ	0.99	0.98	0.99	1
ক	0.99	0.98	0.99	1
খ	0.98	0.98	0.99	0.96
গ	0.99	0.99	0.98	0.99
ঘ	0.99	1	0.99	0.99
ঙ	0.99	0.98	0.94	0.98
চ	0.96	0.94	0.95	0.95
ছ	0.97	0.95	0.97	0.96
জ	0.96	0.94	0.94	0.95
ঝ	0.99	0.96	0.98	0.98
ঞ	0.97	0.97	0.97	0.96
ট	0.99	0.98	0.99	0.99
ঠ	0.99	0.95	0.98	0.99
ড	0.97	0.97	0.94	0.96
ঢ	0.96	0.94	0.93	0.96
ণ	0.98	0.98	0.99	0.98
ত	0.99	1	0.98	0.99
থ	0.96	0.98	0.98	0.97
দ	0.95	0.93	0.94	0.95
ধ	0.95	0.94	0.92	0.96
ন	0.93	0.91	0.93	0.94
প	0.98	0.99	0.99	0.99
ফ	1	1	0.99	0.99

Characters	Inception_V3	ResNet50_V2	VGG-16	Proposed Model
ব	0.99	0.96	0.98	0.99
ভ	0.98	0.98	0.99	0.98
ম	0.98	0.98	0.97	0.97
য	0.96	0.94	0.94	0.96
র	0.99	0.98	0.99	0.99
ল	0.99	1	1	1
শ	0.98	0.98	0.99	0.99
ষ	0.99	0.98	0.98	0.99
স	1	1	1	0.99
হ	1	1	1	0.99
ড়	1	1	1	0.99
ঢ়	0.97	0.97	0.98	0.97
য়	1	0.94	0.99	0.99
ৎ	0.98	0.96	0.98	0.98
ং	0.99	0.99	0.99	0.96
ঃ	0.98	0.96	0.99	0.98
ঁ	0.98	0.98	0.98	0.98

Table 5.2: BanglaLekha Isolated F1 score

Actual Character	Predicted Character	Actual Character	Predicted Character
ঢ	চ	ই	হ
ঊ	ঊ	ব	ধ
ঢ	ঢ়	ই	ঈ
ঋ	ঝ	ব	র
থ	য	য	ম
ধ	ঋ	ঙ	ঊ
য	ধ	খ	থ
ণ	ন	থ	য

Table 5.3: Closely Related Characters(Mistakes Made By Models)

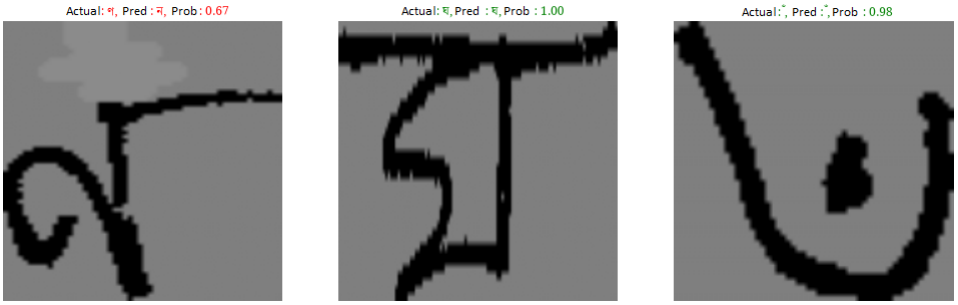


Figure 5.19: Correct predictions by the KDANet on CMATERdb dataset

Figure 5.20: Wrong predictions by the KDANet on CMATERdb dataset

In the table [5.3], illustrates some misdirection's made by the models and the actual characters. Due to handwriting orientation (left vs right) strokes vary and some characters closely resemble each other. Moreover, scanning old documents and converting it into gray-scale produces noise. Furthermore, the noise removal process sometimes accidentally removes dot(.) which is crucial for distinguishing two characters from each other like র and ব ,ঢ and ঢ়. Sometimes due to the cursive nature of the Bangla Language, characters seem the same to machines, which are hardly distinguishable by human eyes.

Chapter 6

Comparative Analysis

We fed our KDANet along with the other 3 models with two different datasets, CMATERdb and the BanglaLekha-Isolated. On the CMATERdb dataset, the ResNet50-v2 model performed the best in training with an accuracy of 98.43%. The Inception-v3 and VGG-16 being second and third, respectively with 98.31% and 98% training accuracy. Whereas our proposed model achieved an accuracy of 95%. However, the Inception-v3 model had the highest test accuracy with 97.73% accuracy followed by ResNet50-v2(96.60%), VGG-16(95.73%) and our proposed model (KDANet) got an accuracy of 94%. On the BanglaLekha-Isolated dataset, the Inception-v3 model achieved the highest training accuracy (98.34%) followed by the VGG-16 (98.08%) and the ResNet50-v2 (97.86%) and our proposed(KDANet) (95.5%) model. All the models performed very similarly during the test phase with the Inception-v3 edging (97.90%) slightly higher than our proposed model (97.66%). The VGG-16(97.40%) and the ResNet50-v2(97.12%) models had the lowest test accuracy of all the models.

	CMATERdb		BanglaLekha-Isolated	
Model	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy
Inception-v3	98.31%	97.73%	98.34%	97.70%
ResNet50-v2	98.43%	96.60%	97.86%	97.12%
VGG-16	98%	95.73%	98.08%	97.40%
Proposed Model	95%	94%	95.50%	97.66%

Table 6.1: Comparison table for training and test phase accuracy

The following bar charts illustrates the results:

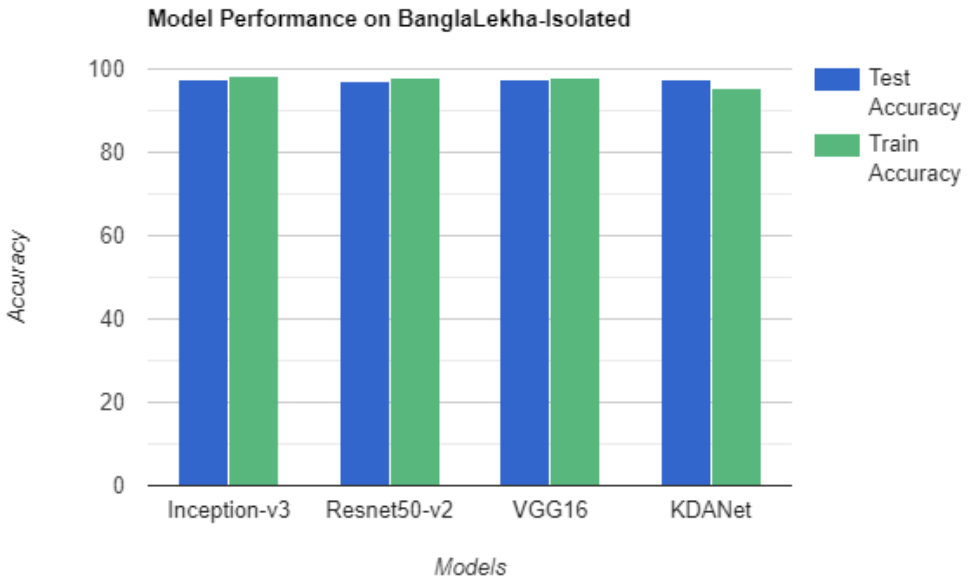


Figure 6.1: Model Performance on BanglaLekha-Isolated

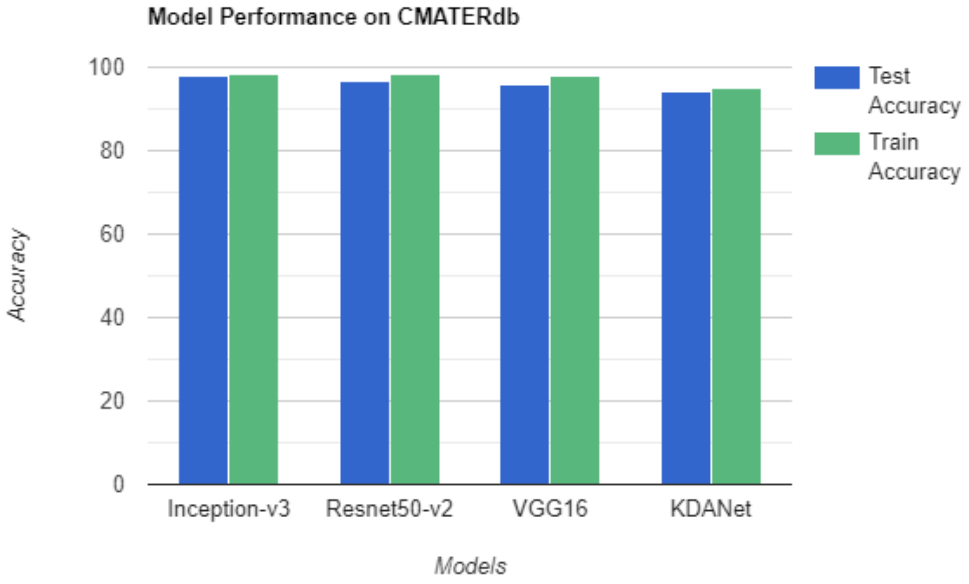


Figure 6.2: Model Performance on CMATERdb

Chapter 7

Challenges

Initially, we wanted to make an OCR for both handwritten and printed data. For lack of availability of synthetic datasets for printed Bangla characters, we had to proceed with only handwritten data. While training our Custom Model , using CMATERdb it resulted in overfitting. To begin with, the dataset was not big enough. So we had to add a dropout layer which eliminated the problem.

Deep learning in general is computational power intensive. Due to unavailability of high-end GPUs and the price hike, we didn't have enough resources to experiment with different models. Tesseract being the leading OCR solution and open source , we wanted to use different features and fuse them with our custom model. Unfortunately, due to complexity of the code we could not do so.

Finally, the resources existing for the Bangla OCR system are not well organized. Also lack of various open source resources to work with have held us back heavily. So indexing them for our many different purposes along the way was troublesome to say the least. It took a significant amount of time to collect resources and previous works.

Chapter 8

Future Works

Initially, we had been planning to build an OCR system which satisfies any sort of Bangla writing. However, the lack of proper datasets proved to be a great barrier in achieving our milestone: especially in the segment of printed Bangla characters. Thus, we aspire to create a synthetic dataset consisting of printed Bangla writings and our plan is to combine all possible Bangla fonts to achieve this feat.

Apart from that, in the near future to work on full sentence based data, the OCR of which is already available for English widely used across the globe. For that, our estimated approach is to make use of Natural Language Processing techniques like segmentation that includes levels such as line level, word level, character level segmentation. To achieve that, we obviously will need to make our model hybrid by combining the strength of our Convolutional Neural Network architectures and merge with BLSTM networks.

In addition to those, we will build a software or app capable of inputting an image and convert the writings there to an editable text format. This too exists for the English language and so our goal is to create a similar type of system for Bangla language. As for our progression in these aspects, we are working currently on those but in an incomplete stage which is not enough to showcase our results for the time being.

Chapter 9

Conclusion

In a nutshell, Bangla is one of the most difficult languages to implement an OCR system due to its cursive nature and structural complexity, not to mention the compound characters. However, we take it as a challenge, as implementation of Bangla Language OCR system will enable many opportunities for not only the Bengali speaking population but also for the people who are interested to travel to this region of the world.

Bangla OCR is still in its primitive stage but we are optimistic that by putting state-of-the-art technologies in harmony with each other, we can produce better results with our custom-made KDANet architecture based model. However, further extensive research is a must for making our research fruitful. The question remains whether the future generations will utilize research materials like ours and improvise accordingly. Still, there is hope as Bangla Language OCR is upgrading more and more with every passing year due to the utmost sincerity displayed by our fellow researchers. Soon the day will come when Bangla OCR will reach the heights of the standard OCR system for English language.

Bibliography

[1] BM Abir, Somania Nur Mahal, Md Saiful Islam, and Amitabha Chakrabarty. Bangla handwritten character recognition with multilayer convolutional neural network. In *Advances in Data and Information Sciences*, pages 155--165. Springer, 2019.

[2] Chandranath Adak, Bidyut B Chaudhuri, and Michael Blumenstein. Offline cursive bengali word recognition using cnns with a recurrent model. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 429--434. IEEE, 2016.

[3] Mujadded Al Rabbani Alif, Sabbir Ahmed, and Muhammad Abul Hasan. Isolated bangla handwritten character recognition with convolutional neural network. In *2017 20th International conference of computer and information technology (ICCIT)*, pages 1--6. IEEE, 2017.

[4] Mithun Biswas, Rafiqul Islam, Gautam Kumar Shom, Md Shopon, Nabeel Mohammed, Sifat Momen, and Md Anowarul Abedin. Banglalekha-isolated: a comprehensive bangla handwritten character dataset. *arXiv preprint arXiv:1703.10661*, 2017.

[5] John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

[6] Ahmed Asif Chowdhury, Ejaj Ahmed, Shameem Ahmed, Shohrab Hossain, and Chowdhury Mofizur Rahman. Optical character recognition of bangla characters using neural network: A better approach. In *2nd ICEE*. Citeseer, 2002.

[7] Muhammed Tawfiq Chowdhury, Md Saiful Islam, Baijed Hossain Bipul, and Md Khalilur Rhaman. Implementation of an optical character reader (ocr) for bengali language. In *2015 International Conference on Data and Software Engineering (ICoDSE)*, pages 126--131. IEEE, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770--778, 2016.

[9] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *arXiv preprint arXiv:2009.12836*, 2020.

[10] Asif Isthiaq and Najoa Asreen Saif. Ocr for printed bangla characters using neural network. *International Journal of Modern Education & Computer Science*, 12(2), 2020.

[11] Mohammad Meraj Khan, Mohammad Shorif Uddin, Mohammad Zavid Parvez, and Lutfur Nahar. A squeeze and excitation resnext-based deep learning model for bangla handwritten compound character recognition. *Journal of King Saud University-Computer and Information Sciences*, 2021.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Mohammed Aarif KO and Sivakumar Poruran. Ocr-nets: variants of pre-trained cnn for urdu handwritten character recognition via transfer learning. *Procedia Computer Science*, 171:2294--2301, 2020.

[14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[15] Farjana Yeasmin Omee, Shiam Shabbir Himel, Md Bikas, and Abu Naser. A complete workflow for development of bangla ocr. *arXiv preprint arXiv:1204.1198*, 2012.

[16] Rahul Pramanik and Soumen Bag. Shape decomposition-based handwritten compound character recognition for bangla ocr. *Journal of Visual Communication and Image Representation*, 50:123--134, 2018.

[17] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55--69. Springer, 1998.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929--1958, 2014.

[20] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm--a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[21] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818--2826, 2016.

[23] Haibing Wu and Xiaodong Gu. Max-pooling dropout for regularization of convolutional neural networks. In *International Conference on Neural Information Processing*, pages 46--54. Springer, 2015.

[24] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818--833. Springer, 2014.