A Deep Learning Approach For Bangla Optical Character Recognition System

by

Farhan Shahoriar
18101682

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

Department of Computer Science and Engineering
Brac University
February 2025

Abstract

Bangla Optical Character Recognition has emerged as a crucial aspect of document digitization. A wide range of sources are covered, such as computer-generated text, letterpress, typewriters, outdoor banners, posters, and handwritten documents sourced from diverse channels. Bangla character structure is very complex, handwritten documents pose a significant challenge in recognition. To overcome those challenges, we introduce the custom Convolutional Neural Network model using image classification to detect and recognize the characters. This model can learn from a large-scale Bangla handwritten dataset. BanglaLekha-Isolated[15] and Eksush[19] datasets are tested in our model. To compare the performance of the custom Convolutional Neural Network model, we used the two most popular deep learning model, ResNet-50 and VGG16. By implementing our model, we achieve 95.63% accuracy on Ekush dataset and 96.20% on BanglaLekha-Isolated dataset. Bangla OCR research field is not saturated and also challenging due to its character style is very much complex compared to other language. To conclude, ongoing research efforts are dedicated to advancing Bengali OCR technology, making it more reliable and accessible for users. As with any OCR system, the quality of input images and variations in writing styles remain critical factors, but progress is being made to overcome these challenges.

Keywords: Bangla handwritten-character recognition; Deep Learning; Convolutional Neural Network; Residual Block

Acknowledgement

Firstly, all praise to the Great Allah for whom my project have been completed without any major interruption.
Secondly, to my advisor Mr.Annajiat Alim Rasel sir for his kind support and advice in my work. He helped me whenever I needed help.
And finally to my parents without their throughout support it may not be possible. With their kind support and prayer, I am now on the verge of my graduation.

Declaration

It is hereby declared that

1. The project submitted is my original work while completing degree at Brac University.
2. The project 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 project does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I have acknowledged all main sources of help.

Student's Full Name & Signature:

Farhan Shahoriar

18101682

Approval

The project titled "A Deep Learning Approach For Bangla Optical Character Recognition System" submitted by

1. Farhan Shahoriar (18101682)

Of Fall, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on February 3, 2025.

Examining Committee:

Supervisor:
(Member)

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

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
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

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

ANN Artificial Neural Network

CNN Convolutional Neural Network

$Grad - CAM$ Gradient-weighted Class Activation Mapping

$LSTM$ Long Short Term Memory

OCR Optical Character Recognition

$ReLU$ Rectified Linear Unit

$ResNet$ Residual Network

RNN Recurret Neural Network

VGG Visual Geometry Group

Chapter 1

Introduction

Optical character recognition (OCR) is used to electronically transform scanned images, documents, into machine-readable text which can be edited or stored digitally. It has a vast area of working opportunities on office automation, converting books into PDF, maintaining newspaper archives, human to computer interaction etc. With the help of OCR data, can be easily accessible, which helps with data entry, searching, translation etc. OCR helps us to get relief from this tedious, monotonous work and saves our valuable time. In the current scenario, a numerous amount of research with exceptional outcomes has been done on English, Latin, Arabic, Persian, Chinese, Japanese and Korean etc. languages. The Bangla language originates from the Indo-European language family. From its branch Indo-Aryan language family, Bangla language eventually evolved. Bangla is the world's seventh-largest language and the second most used language in the Indian subcontinent. Over 300 million people speak in Bangla as it is the main language in Bangladesh and widely spoken in West Bengal, Tripura, and Assam of India. The Bangla language consists of 50 basic characters, including 11 vowels, 39 consonants, and 10 numerals. The Bengali script is complex with its left to right writing direction with a combination of consonants, vowels, diacritics and conjunct consonants. Bangla Handwritten characters are difficult to recognize due to various writing styles, cursiveness and compound characters. Bangla OCR is a challenging task as it requires both character detection and recognition in a single time with its unique characteristic. The domain of Bangla OCR achieve a significant amount of research for the advancement but it is not sufficient. There's a large amount of potential for Bangla OCR development which is yet to be done. In this project, we develop a custom CNN model to implement Bangla OCR.

1.1 Background

Bangla is one of the most intricate scripts to write. Bangla Handwriting varies from person to person, which is a great challenge for the machine learning model to understand. A numerous research has been done on this domain, but there are some drawbacks which need to be fulfilled to develop a complete Bangla OCR system. Due to automation in the workplace, people have to fill various forms and computerize various documents, which can be solved by Bangla OCR.
First and foremost, recognizing the convoluted edges of different characters is a challenging task. Bangla script uses more than 300 conjunct consonants, which can

be combined with two or more characters and develop a unique shape. Additionally, consonant and vowel modifiers, diacritics which can be placed up, down and between a character and change its pronunciation. It can be challenging to detect those modifiers and diacritics. A header stroke is also used to write words called 'Matra' which also needs to be detected as some characters don't have header stroke. Moreover, there are not enough annotated datasets which cover all the quality to conduct further research. In addition, those datasets have noise and distortion in some images. That's why it needs to treat each data individually, which consumes a lot of time. Furthermore, the fonts are diverse in printed and handwritten text. Overcoming those challenges, there are some unpredictable problems which needs to be addressed during research.

1.2 Motivation

In this project, our goal is to develop a custom CNN model and deploy on Bangla large-scale datasets. As most of the deep learning models were introduced for the English language. Our main purpose of the research is to correctly identify the handwritten Bangla Characters. For our model, we need to understand the Convolutional Neural Network architecture and also need to implement a pre-trained model to our data set for compare performance with our model. With trial and error, we aim to build a refined model to handle Bangla complex characters. After training our model, we implement it to build a web application using the Flask web framework.

Chapter 2

Related Work

Bangla OCR is important for our everyday life and also for our history and culture. By implementing OCR technology in professional work we can save time and money, also we can reduce carbon footprint by using less paper. Early in 1950, research on OCR technology emerged. By early 2000, Bangla OCR had been developed with many possibilities. Early researchers faced various complexity due to the limitation of resources. Now with the modern deep learning method, the researcher has achieved a decent amount of progress in this domain.

The [6] authors used a feature-based method of Bangla OCR. In preprocessing, they used a thresholding approach with histogram to convert the digitized image into two-tone. For skew detection they observed the headline of Bangla script, if they find any error angle, then they solve it by a quick skew algorithm. Bangla text line can be divided into three distinct zones, line segmentation is done after it scanned vertically for character and word segmentation. For character segmentation, they only consider the middle zone and rub off the headline so that each character get disconnected. Researchers classified the compound character in a subgroup for faster and accurate detection of basic characters. They used a combination of feature based and template matching approach for compound character recognition. After all the preprocessing stages, a feature-based tree classifier was used to group all characters into smaller clusters. Subsequently, a template matching method was applied to recognize each group individually. An error detection along with correction mechanism was fully deployed at last. Furthermore, they achieve 95.50% accuracy in word level after implementing all of this.

In [10] researchers find that Bangla handwritten digit recognition presents a very meaningful problem in the area of computer vision because of the natural and normal variation in handwriting styles. Handwriting character recognition is a critical task, as every human has their own writing style. They used a deep Convolutional Neural Network to achieve their desired outcome. For this, they use numtaDB dataset which is an augmented, unbiased, along with unprocessed dataset. This dataset is a combination of six datasets gather from different sources. For preprocessing, they resize each image into 32×32 pixels from 180×180 pixels and convert images into gray scale. Due to resizing, lost image data retrieved by inter-area interpolation method. Gaussian blur was added first, then the blurred image was used to lessen from the original and increase edge details. After a 3×3 matrix was applied using a

Laplacian filter and a median filtering was applied to remove all the salt and pepper noise. For more specified image, a segmentation was applied using Otsu's method for global thresholding. The suggested CNN design has a total of six convolutional layers, followed by a pair of fully connected dense layers. The Rectified Linear Unit (ReLu) was employed as the activation function, max pooling and batch normalization was used after every two layer. For the output layer, softmax activation function is used. They used Adam optimizer to update weights. After applying this custom CNN, they achieve an unweighted average accuracy of 92.71% after 30 epochs on numtaDb dataset.

According to [5] a feature based approach with tree classifier to recognize the basic character. The researcher initially digitized the characters, and then used a histogram-based thresholding method for converting the images into a two-tone format. For detection of skew, they suggested a helpful method through detailed analysis of the script's headline. Connected components whose bounding boxes are greater than the average box width of the components are selected for skew detection. For word and character segmentation, they plotted a histogram on the horizontal axis. Before going to actual character recognition, they grouped the characters into three subgroups. After all these preprocessing, the feature based tree classifier was used. For some characters, another heuristic was implemented close to the tree's leaf node. Regarding compound characters, a feature-based tree classifier was first used in order to categorize them within tinier groups. Then, a template-matching method with adequate efficiency was readily employed in the identification of individual characters. After implementation of error detection as well as correction, researchers found that a majority of rejected characters were compound characters. Most researchers found this result. Since the classifier employed a tree structure, one could identify a test character's subgroup before rejection, letting error correction be restricted to only that subgroup.

At [9] the authors emphasize on Long Short Term Memory model for OCR. There are two main approach for OCR. One is segmentation and other one is without segmentation based approach. Segmentation approach require well-designed segmented model. Since segmentation errors, it leads to error in output. Segmentation based OCR needs careful estimation of classification cost. In particular, it may require a heuristic approach to tune the cost functions. Here are the authors talks about the drawback of segmentation free model as Hidden Markov Model. This model requires cautious selection of the model's construction, and it faces challenges in heuristic estimation to their cost function to get a decent performance. Another model as Convolutional Neural Network used to detect handwriting character recognition problems with positive outcomes but have obtained lower performance compared with segmentation. Recurrent neural network doesn't have much performance due to vanishing gradient problem. To mitigate this issue, Long Short Term Memory was introduced. In this paper, the authors emphasize on LSTM architecture and found better performance than other models. For text line normalization, it detects the lower line and the x height of the text. After they applied 1d bidirectional LSTM architecture. Training was achieved by implementing a forward propagation through LSTM and output, followed by a forward-backward alignment for the output with ground truth and lastly a back propagation was implemented. They use University

of Washington's (UW3) datasets for English input and achieve a 0.6% error on the test set. For Fraktur dataset they achieve 0.16% error on the test sets.

In [8] the authors use Tesseract OCR engine for Bangla OCR. To implement this, the training data consists of 340 characters with different combinations of data using different parameters. After pre-processing the image, they prepared Tesseract supported image, as it's only able to read an uncompressed 1 bit/8 bit tiff format image. Implement Tesseract engine on this preprocess image and generated an output for post-processing. Post-processing involves in two phases. In the initial phase, it captures the Unicode text and address the known error. On the second level, it performs a spell checking operation. For clean printed document they achieve a 93% accuracy, 85% for printed book & newspaper and 70% for screen print image.

In [20] the authors prepare Bangla Script Recognition support of Tesseract engine. For this, they first generate training data. The training data following combinations of all vowels, consonants, numerals, consonants & vowel modifiers and conjunct consonants. Later they typed all the combinations and processes the image. Each training image uses a box file. To generate box file they manually refined the transcription in the created box file of Tesseract. All the images have to match the box file must match the training unit of the image. Tesseract uses its own tool to create box information. The authors performed clustering to create the prototype using the character shape feature.To create character set, they used unicharset_extractor. Tesseract uses 3 dictionary file for each language. After creating the dictionary.They also identified possible intrinsic ambiguity between characters. For test data processing they use binary thresholding, noise reduction, skew detection and correction, character segmentation. Implementing this data set into tesseract, they achieve a 98% accuracy in recognizing basic characters and 92% accuracy in page images with same and different fonts. In this paper, the authors show the complete procedure to create Bangla OCR datasets for tesseract OCR engine.

The authors[18] shows how to collect and process handwritten data for Bangla language. First, they create a form with equal size cell. They label each box to write a specific character. After filling the form by volunteer, they scan those forms at 300 dpi. For skew in paper, they create a black boundary around the paper and used an algorithm to find the skew angle and crop the images on the edge of the black bounding box. After they crop each image into row and column wise which separated each character. Thresholding is applied for noise removal of images using Otsu's algorithm for optimal thresholding value. Then Gaussian blur was applied for image smoothing. The authors use a custom algorithm for removing extra background of the images. Initially, all images will be in inverted form. To sum up, in this research the authors introduce us a generalized way for preprocessing any handwritten character database.

Chapter 3

Methodology

For character recognition, we used a custom Convolutional Neural Network model for Bangla Character. A CNN model is commonly used in the field of computer vision. CNN is great for image classification and object detection, it can extract image features and learn from them. The main building blocks of CNN are the Convolutional layers. We used BanglaLekha-Isolated and Eksuh dataset to deploy in our model. ResNet-50 and VGG16 models are used for comparison.

3.1 Dataset

Bangla OCR dataset is not as widely available as English dataset. Recent research is ongoing in the field of generating a good Bangla OCR dataset. Also, most of the dataset does not have support for compound characters. A few datasets have promising outcomes on this problem. BanglaLekha-Isolated is one of the renowned dataset consisting most of the compound characters, but it doesn't have the support for modified vowels. To overcome this, Eksush Data set plays a vital role for OCR research by providing most of the possible support for Bangla OCR dataset. For training our model, we used BanglaLekha[15] isolated and Eksuh [19] dataset. BanglaLekha-Isolated 3.2 has 1,66,105 handwritten images with 84 classes. These classes consist of 50 basic Bangla characters, 10 numerals and 24 compound characters. This dataset also contains age and gender of the subjects from whom the samples were collected.The Ekush dataset 3.1 has 673,482 images. It has 50 basic Bangla characters, 10 digits, 10 compound modifiers and 52 compound characters.

3.2 Data Preprocessing

Data pre-processing, one of the most important steps for deep learning. Data plays a vital role to train the model. Firstly, we resize the images according to our need. As ResNet-50 and VGG16, expected image size is 224 x 224 pixels. Ekush dataset has by default 28x28 size image and BanglaLekha Isolated hasn't fixed size image. It can be used mostly for all the deep learning model to their required input size. After resizing, we apply data augmentation. Data augmentation can improve model performance. It includes transformation, rotation, height shift, width shift, horizontal flip, color adjustment etc. Data augmentation only applied in training data for better learning. Bangla is very much complex character, if we over do

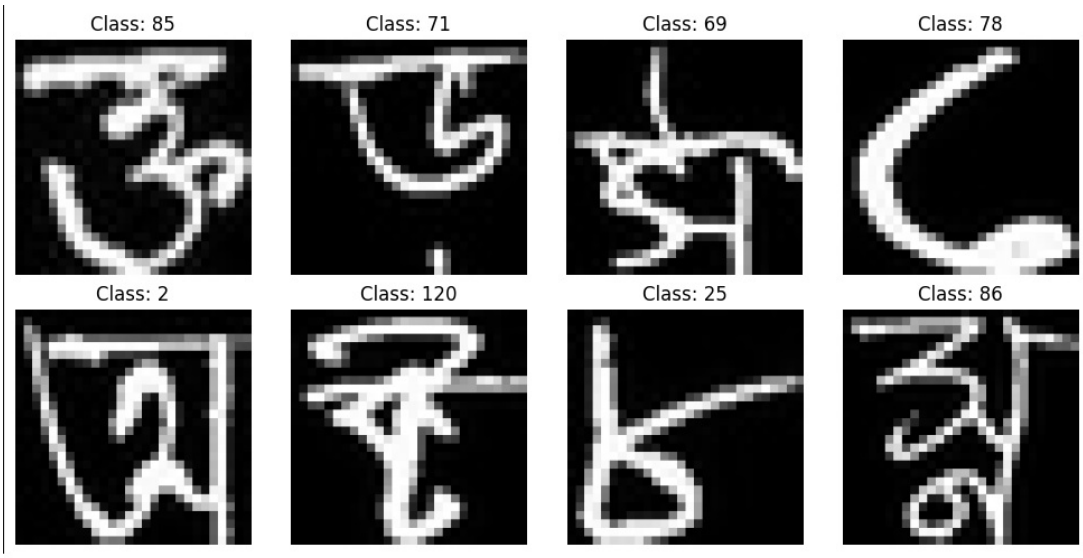

Figure 3.1: Samples from Ekush Dataset

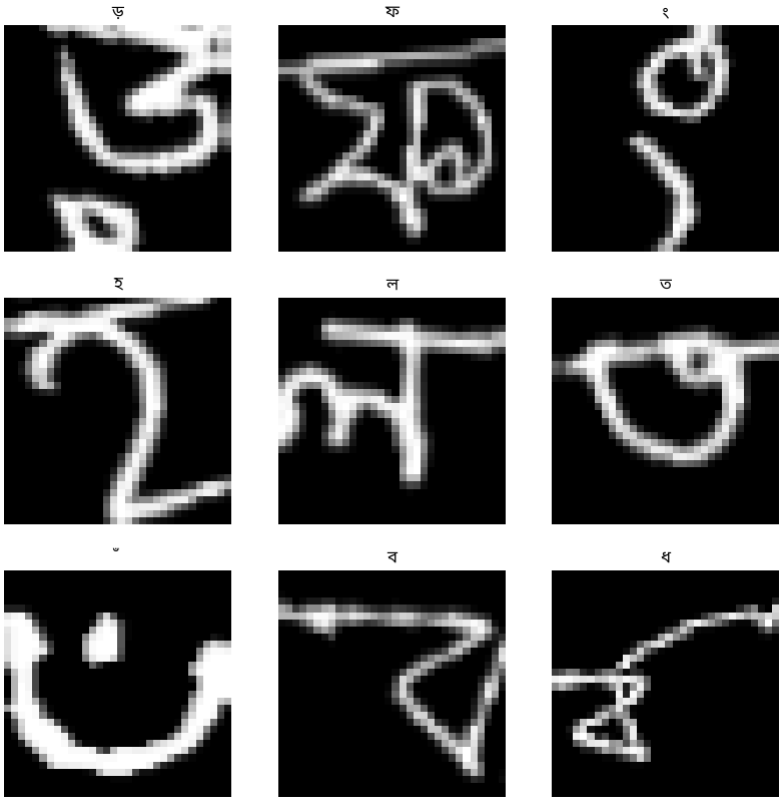

Figure 3.2: Samples from BanglaLekha Isolated Labeled Data

data augmentation we can lose some characteristic of the data. For our model, we rotate the image 5% and height width shift by 5%. Images are gray scale which contain pixel value 0-255. We normalized the images to 0-1 using rescale in data augmentation. After, we split the dataset to 80 and 20 ratios for training and validation. For BanglaLekha Isolated dataset, we got 84 classes as labels and 122 classes as labels for Ekush Dataset.

3.2.1 Image Processing

For our web application, we need to take input from the user. After, we resize the image by setting the width to 600 pixel and height by using aspect ratio. Then the image convert into grayscale to reduce computational complexity. Later, we use following pre-processing technique.

Noise Removal: Images often contains noise due to various reasons. Using scanned images or printed images is often not of good quality. Also taking images from different camera, which may impact image quality. To process images for recognition, we need to filter noise. If we do not reduce noise in our image, it can affect recognition. For this, Gaussian Blur [4] is applied. It uses a filter to remove noise from the image. Gaussian function assign weights to the kernel and uses normal distribution to each pixel of an image.

Thresholding: For contrast an image, we use threshold using binarization. There are two types of thresholding one is adaptive and other one is global. In adaptive thresholding[7] it uses local binarization technique which compute the threshold value which varies across the image pixel. Gaussian weight is used in this. Global Thresholding use a single value across the image to detect foreground and background. Otsu's[1] method is the most effective in global thresholding. It uses a histogram to calculate the intensity of each pixel. After calculating probability distribution and class variance, it selects the optimal threshold value. This is the best technique to separate the foreground and background.

Dilation: It is used to enlarge the boundary region of an image in binary or gray scale format. It uses a kernel to extract the shape of the image and determine the dilation using mathematical morphology. When the kernel pass through the image it matches the foreground pixel, if it matches then it maximized the neighboring value. Dilation[2] is used for contour detection, connecting objects and filling holes in image.

Canny Edge Detection: The canny edge detector is useful to extract relevant edges from an image. It is very useful in the field of computer vision and image processing. Canny's algorithm[3] is developed by John F.Canny in 1986. The algorithm uses a multi-stage process. First, a Gaussian filter is applied to remove noise from images. After that, it finds the intensity of the images using gradient. Followed by gradient magnitude, thresholding is used to maintain a true edge and double threshold to detect strong edges. Finally, track the hysteresis to restore the weak edges connected to the strong edges.

Region Of Interest(ROI): ROI extract a part of image for further operation. For OCR, it is selected for individual processing and classification of an image. ROI is defined by a boundary, which can be any shape. It uses contour detection technique to detect objects in an image. The bounding box is calculated by x, y, w, h. Using ROI, the box images are used for further classifications.

3.3 Proposed Model

In this study, we introduce a deep learning method for recognizing Bangla characters. The model uses a custom Convolutional Neural Network (CNN) with residual block to extract relevant image features and accurately classify them as Bangla characters. A considerably improved Residual Convolutional Neural Network, optimized for recognizing handwritten Bangla characters. Used numerous Convolutional Layers to extract features, six strong skip connections and two effective regularization techniques are combined to address important character complexity, large noise and large dataset variability. In the first two layers, we used 64 size filters with 3x3 kernel followed by Residual block. We used max pooling to reduce feature size. ReLU activation function is used for solve non-linearity. Conv2D layer followed by batch normalization. After the Convolutional layers, a global average pooling layer summarizes each image's feature maps into a single vector. After global average pooling, a 256-unit dense layer, followed by a final dense layer that outputs Bangla characters. To prevent over-fitting, dropout layers have been implemented.

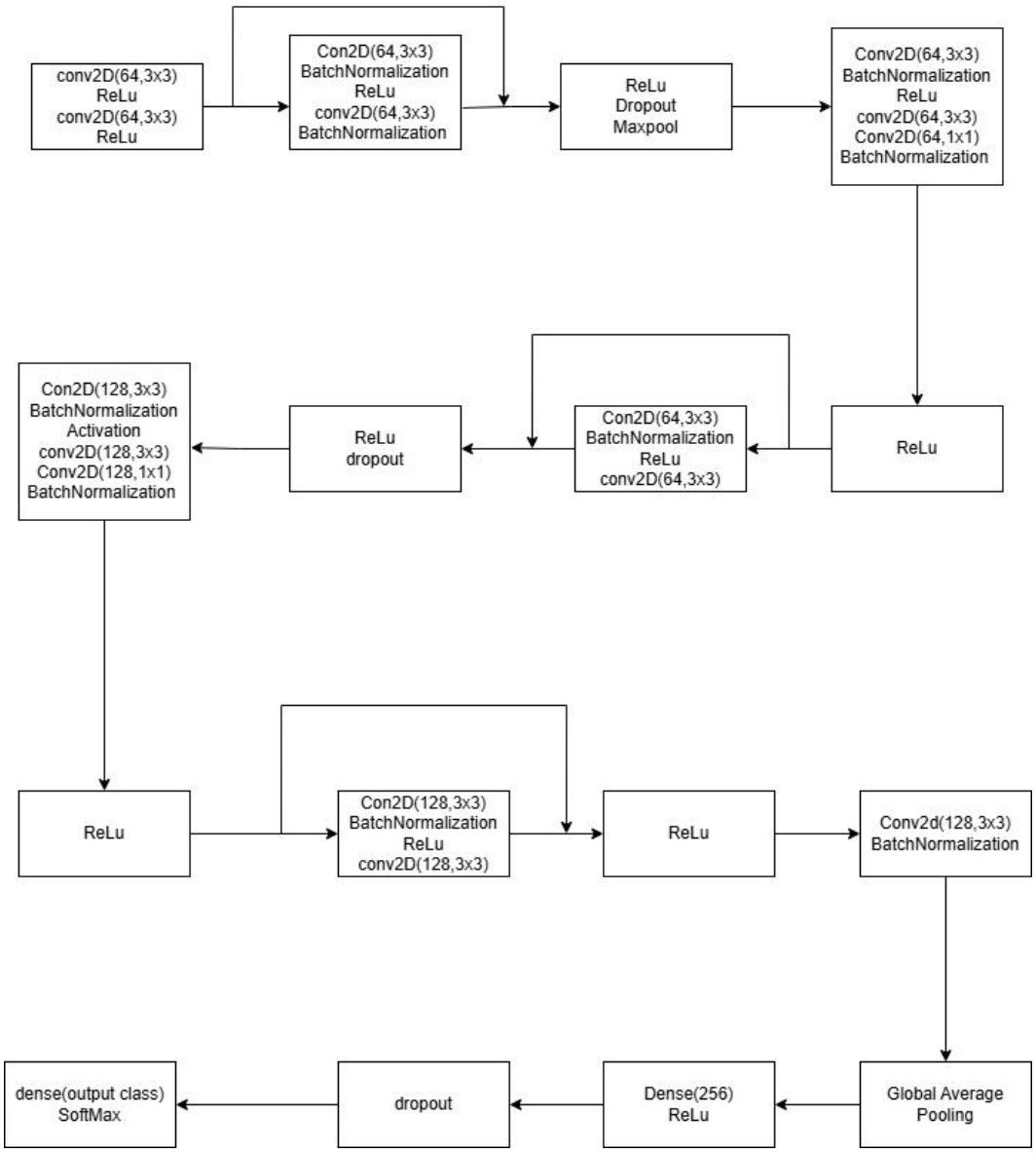

Figure 3.3: Proposed CNN model

Residual Block: When the network gets deeper, the accuracy degrades due to optimization difficulties. Using residual block[12] we can map the input layer to the output layer using shortcut connection3.3 bypassing one or more Convolutional layers. This shortcut or skip connecting does not add any extra parameter, so the computa-

tional time does not increase. Gradient is used to update the network weights using chain rule calculation as the network gets deeper, the gradient becomes effectively vanishing. To encounter this issue, residual block helps by allow the gradients to bypass the layers during back propagation and flow strongly. These block consist of multiple Convolutional layers followed by batch normalization, ReLU activation function. In the first Convolutional layer it uses 3x3 kernel to extract features followed by batch normalization and ReLu activation. The Second Convolutional layer follows the same without the ReLu activation. It also adjusts the shortcut connection to match the output dimension when needed.

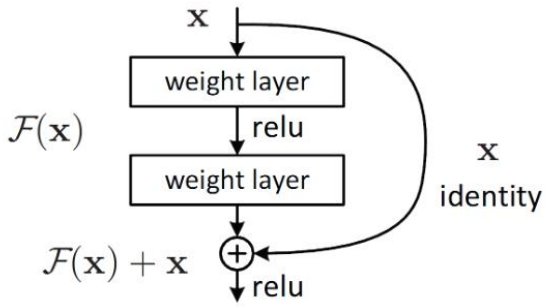

Figure 3.4: Residual Block

$$\mathcal{H}(x) = \mathcal{F}(x) + x \tag{3.1}$$

Batch Normalization: Batch Normalization used to train model faster and more stable. Batch Normalization layer gets inserted between a hidden layer and the next hidden layer. The main purpose of this is to normalize the previous hidden layer output before passing it into the next hidden layer. First, it computes3.3 the mean and variance of the batch. Then normalized the activation by ensuring that activation has zero mean and unit variance. Then uses scale and shift to the normalized activation3.3 to learn optimal scaling and shifting.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3.2}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{3.3}$$

Global Average Pooling: Designed to replace the fully connected layer. It takes average of each feature map resulting a single value per feature map and pass directly to the soft-max layer. This helps reduce the complexity of the model.
ReLU: Rectified Linear Unit is an activation function. The activation3.3 function always retains the maximum positive values from the output and neglect the negative value and convert them into 0. This helps the neural network to learn complex patterns more efficiently.
Softmax:3.3It is ideal for multi class classification using probability distribution. This function calculate the probability of each class. The output of the softmax

ensure that the sum of all probabilities equal to 1.

$$\sigma(\vec{z})_1 = \frac{e^x}{e^x + e^0} = \frac{1}{1 + e^{-x}} \tag{3.4}$$

$$\text{RELU}(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x >= 0 \end{cases} \tag{3.5}$$

3.3.1 VGG16

Introducing VGG16 [23] with 16 layers, including 13 convolutional neural networks with 3 connected layers. The network is composed of convolution layers of 3x3 convolutions with stride, padding and 2x2 max pooling with stride 2. This model is best for feature extraction. The feature increased after first two conv2D layer. It changes constantly. It is good for recognizing curves, line and shape. There are three dense layers, the first two layers with 4096 neurons and the last layer we use as per our output class. ReLu activation layer used in all the convolutional layers except the final layer which uses softmax activation function. For Ekush dataset we have to resize the image into 32X32 pixels as by default it was 28x28 pixels. BanglaLekha Isolated has large size data for implementing this model in the dataset we resize 10 vowels value from the dataset by 224X224 dimension This model achieves a better accuracy for a large scale data with cons with high computational cost.

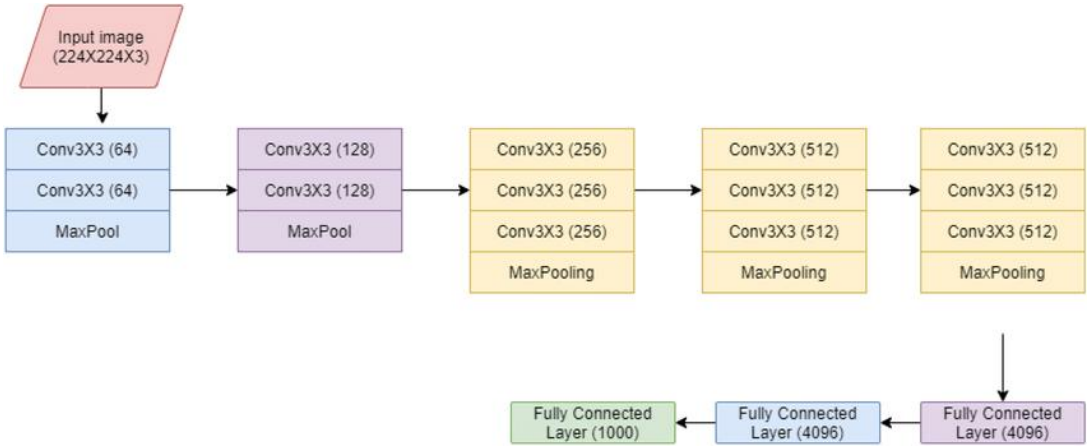

Figure 3.5: VGG16

3.3.2 ResNet-50

The main idea of ResNet-50 [12] is based on residual learning, which enables deep learning network to train more effectively. Residual block is used for skip connection. To solve vanishing gradient problems, it introduces skip connection by going into a deeper network. It has 4 stages of residual blocks, which contain multiple Convolutional layers. For the model we take a batch size of 32 with only 10 classes from both of the dataset. We resize the image into 224x224 pixels and invert the gray scale image and change it into a RGB channel as by default Resnet50 takes 3 channel input. Here it uses an identify function to stop the next layer for worsening the performance. We implement this model in Ekush dataset, taking only digits. It provides good feedback with large deep networks, unlike other models which can worsen the performance.

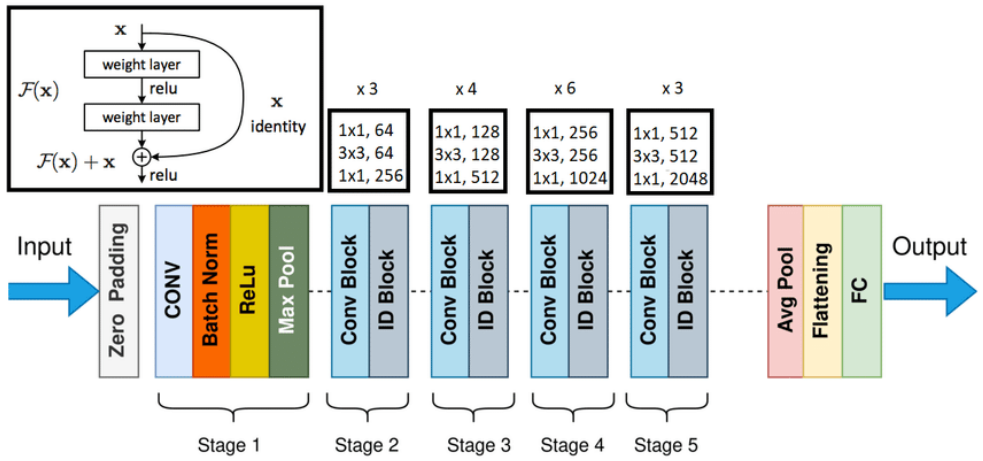

Figure 3.6: ResNet-50

The table 3.1 shows different work approach on Bangla OCR. [16] used a hybrid hog model with 2,477,138 parameters and achieved 83. 69% validation accuracy. The authors show different hybrid modes for Bangla characters. A 3:1 ratio used in training and validation on BanglaLekha-Isolated dataset with 50 basic characters. The hybrid model consists of ANN and CNN. For feature extraction, the authors used a histogram of oriented gradients(HoG). At [14] the authors used deep CNN with dropout regularization and exponential linear units in CamterDB 3.1.1 dataset with 171 class. Using ELU helps to bring gradients closer to natural gradient. To prevent overfitting dropout, layers being implemented to force the network to learn diverse features. This model achieves a 93.68% accuracy. [24]InceptionResNetV2,enseNet121 and InceptionNetV3 applied on CamterDB 3.1.1 dataset achieve

Reference	**Model**	**Accuracy**	**Dataset**
Sharif et al.,[16]	Hybrid-HoG	83.69%	BanglaLekha-Isolated
Ashiquzzaman et al.,[14]	DCNN with ELU and Dropout	93.68%	CMATERdb
Azim et al.,[11]	SVM	83.68%	BanglaLekha-Isolated
Rabby et al.,[17]	CNN	95.71%	BanglaLekha-Isolated
Alif et al.,[13]	ResNet-18	95.10%	BanglaLekha-Isolated
Srizon et al.,[25]	CNN	96.09%	Ekush
Azad et al.,[21]	Deep CNN	95.21%, 95.53%	BanglaLekha-Isolated, Ekush
Ghosh et al.,[24]	InceptionResNetV2, DenseNet121, InceptionNetV3	96.99%, 96.55%, 96.20%	CMATERdb
Asraful et al.,[26]	CNN	92.48%	BanglaLekha-Isolated
Proposed Model	CNN with Residual Blocks	95.63%, 96.20%	Ekush, BanglaLekha-Isolated

Table 3.1: Comparison between previous work with proposed model

above 96% accuracy, but this model has large parameters compared to our proposed model. Our model consists of total 1,005,050 parameters which reduce the computational time. Using the concept of residual block with skip connection, we achieve 95.63% accuracy on Ekush dataset with 122 class and 96.20% accuracy on BanglaLekha-Isolated dataset with 84 class. Our model outperforms the CNN based approach for Bangla character classification

Chapter 4

Implementation

4.1 Setup

For this project, we used python programming language and TensorFlow library. TensorFlow library is one of the most recognized among deep learning library. Later, we used flask web framework to build a web application using the pre-trained custom CNN model to recognize Bangla handwritten character. Flask is a micro web framework written in python. For training our model we used a computer with a 10th generation Intel Core i7-10510U Processor, up to 4.90 GHz with 8 GB Ram.

4.2 Workflow

A CNN model with residual block3.3 is used in our model. The model takes an image as input of 28 X 28 pixels. In the first layer, it accepts single channel normalized gray-scale image. The model consist of multiple Conv2D layer using residual block with a kernel size 3x3 to detect features from Bangla character. After the conv2D layer, batch normalization is applied with ReLU activation. By using residual block, we use a technique called skip connection or shortcut. The shortcut connection skip some layers in between and added other deeper layer in the block. We take the input layer and subtract from the output layer, later we add that using skip connection.3.3 This helps the model to learn better during back propagation. Each residual block ReLU activation3.3 and max pooling layer is used. After first two conv2D layer with 64 filter a residual block is implemented followed by ReLU activation, dropout with 0.2 and max pooling layer. Next a Conv2D layer with 64 filters with batch normalization added with a residual block only with a ReLU activation with another residual block. A 0.2 dropout added with 2 residual blocks with a conv2D layer. Before global average pooling, a conv2D layer with 128 filters is used to preserve feature. At final layer, we used two dense layers with, first 256 dense layers after global average pooling. In the output layer, a soft max activation3.3 function added with corresponding to Bangla Character class. We used Adam W optimizer to compile our model. It uses weight decay and applied directly to the weights before update. This is used to reduce over fitting by penalize the loss function. This make the model to generalize well on unseen data. To train our model reducing learning rate is used, it helps the model converge smoothly during training. Reduce on plateau, reduce the learning rate when validation loss doesn't improve. For training, we used a 128 size batch and 30 epochs.

4.3 Web Implementation:

First, we install necessary library for our web framework. Then upload the CNN model in the project folder. We used HTML and CSS to build the fronted of the web application. After we take user input from the user. The image then use a pre-process function to resize the image and set the new width to 600 pixels and determine the new height by the aspect ratio to without distortion. Next, we convert the image into grayscale for better feature extraction with less computational time. Images may have noises due to different lightning condition and those need to addressed. For noise removal, we used a local threshold using Gaussian blur and adaptive thresholding to improve contrast between the background and character in the image. Next, we invert the image, making the background black and the character white. Dilation is applied to make the character bold. For contour, canny edge algorithm detect edges in the image. Subsequently, ROI is applied to the image. Consequently, we make process box for each character recognition using region of interest(ROI), Otsu's threshold and resized the box as per our model input size. Normalizing the image is equally important for our CNN model to predict. Then use the model to predict the character. Additionally, we display the bounding box image in the front-end to debug more easily.

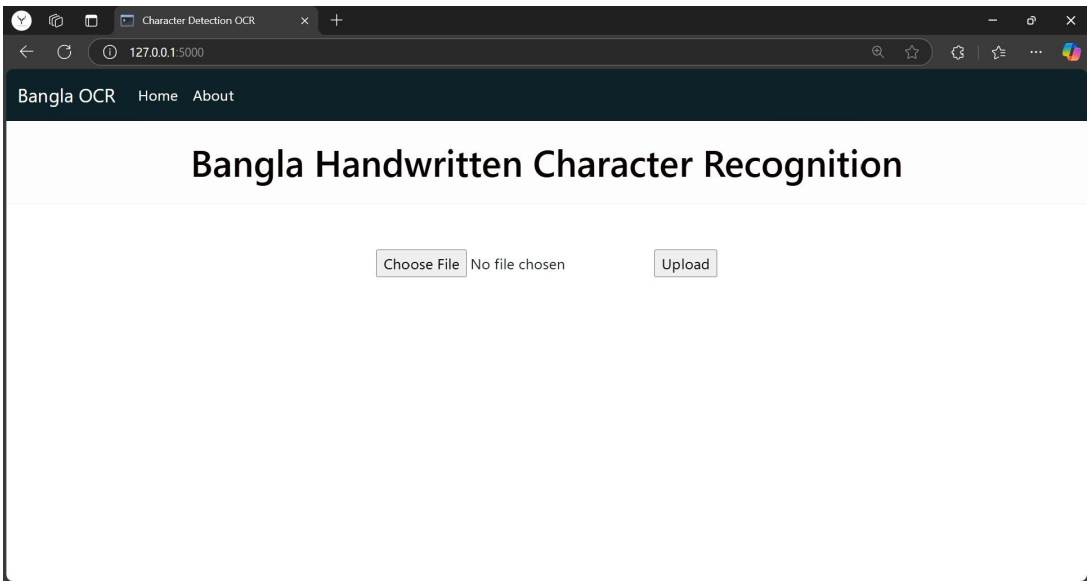

Figure 4.1: Web Application User Interface

4.4 Grad-CAM visualization

By using Grad-CAM[22] we can know the which part of the image contributed most for prediction. It uses gradient flow to analyze the last layer of the CNN

model, generating a heatmap map highlighting the most important part in the image. This helps the model understanding how it's making the prediction, debugging, enhance the transparency. For Grad-cam visualization we used our proposed model trained on Ekush dataset and for prediction we used BanglaLekha-Isolated dataset. 4.2Shows the heatmap on the BanglaLekha-Isolated datasets character presenting the most focused parts of the images while the model makes predictions. The red, yellow and green regions are the most important for the model to predict. These regions consist the most distinctive feature of the images.

Figure 4.2: Grad-CAM Visualization

Chapter 5

Result & Analysis

5.1 Result Analysis

In our project, first we used Ekushdata set to train our model without residual block. Where we achieve 91.47% accuracy. After we used residual block in the same model without changing other factor, we get an 95.63% accuracy.

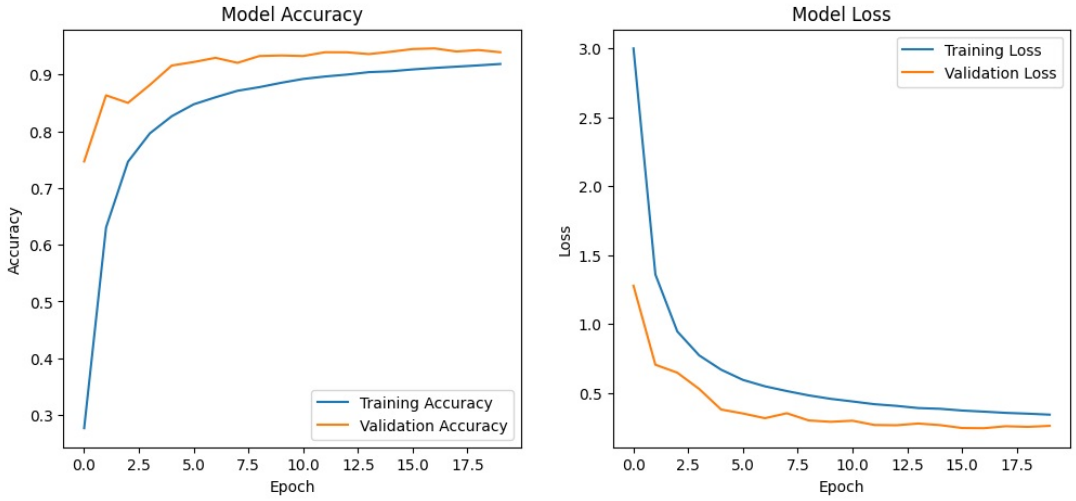

Figure 5.1: Accuracy on Ekush Dataset without Residual Block

The following figures show accuracy and loss curves, and the gap between the curves indicate that the model doesn't over fit. Later we take only the consonant from BanglaLekha-Isolated dataset and resize the images by 32X32 pixels and apply our model with residual block and get accuracy 96.08% with a loss 15.91% validation accuracy: 94.90% validation loss 22.50%. To train our model, we use a 128 size batch with 20 epochs. After 10 epoch, we can see the model perform very well. here

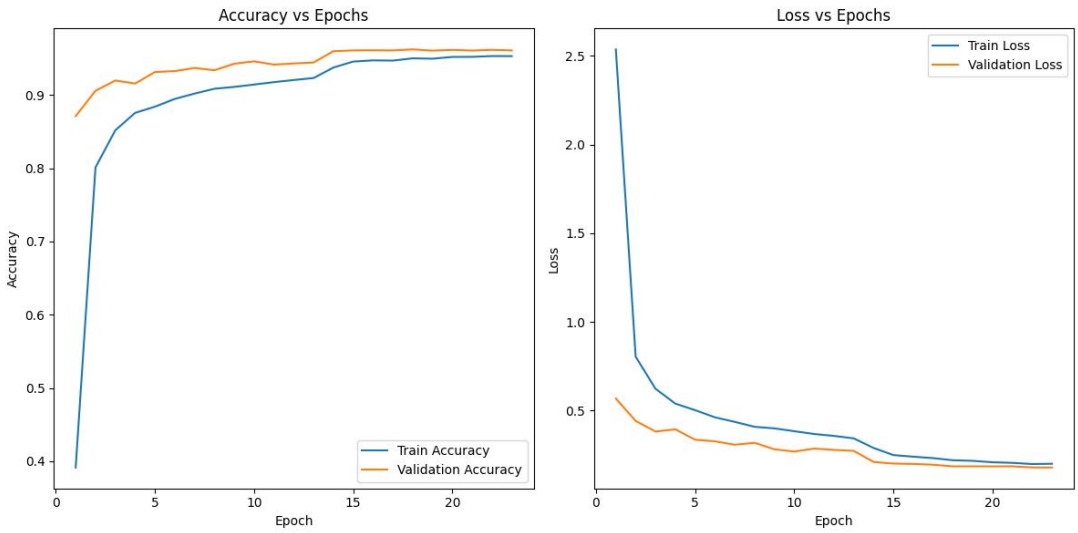

Figure 5.2: Accuracy on Ekush Dataset with Residual Block

we encountered a little under fitting issue. To test our model later, we implemented Ekush dataset digits. For this, we take a 28 by 28 size image. After 20 epochs we got 92.71% accuracy with 28.58% loss with validation accuracy 95.40%. From the curves we learn that if we take lager size image it can perform we with the model as it tends to extract the features more efficiently. As our model struggle with Ekush dataset because the size was 28 by 28 pixels by default but performs well with BanglaLekha Isolated dataset. For ResNet-50 and Vgg16 we used BanglaLekha-Isolated dataset as both the model expect a 224 by 224 size image. Vgg16 got 79.30% accuracy after 20 epochs, after we train the model for additional 12 epochs and finally got 85.72% accuracy. Using ResNet-50, we got 92.70% accuracy with 25.81% loss.

Confusion matrix is used for analyzing model performance. It helps us to visualize the model errors. The following confusion matrix from BanglaLekha Isolated dataset with consonants identified how many images it correctly interpreted with true label and predicted label. We can see the true label is ঢ but the model predicted চ, true label was ভ predicted ড also it struggle with ন and ণ . We can see those characters only distinguished by some matra and some lines. This model trained on handwritten dataset as it difficult to get all the features right because of diverse handwritten styles.

From performance matrix we can see our model face some difficulties to recognize some characters such as “ঋ”, ”ঞ”, ”য” etc. Most of the cases those letters has similarities with other characters. From,5.9 we see that ণ misinterpreted by প. Bangla character is very complex due to handwriting because everyone has different style. Due to this stokes vary sometimes, both the characters hasn’t any matra strokes.

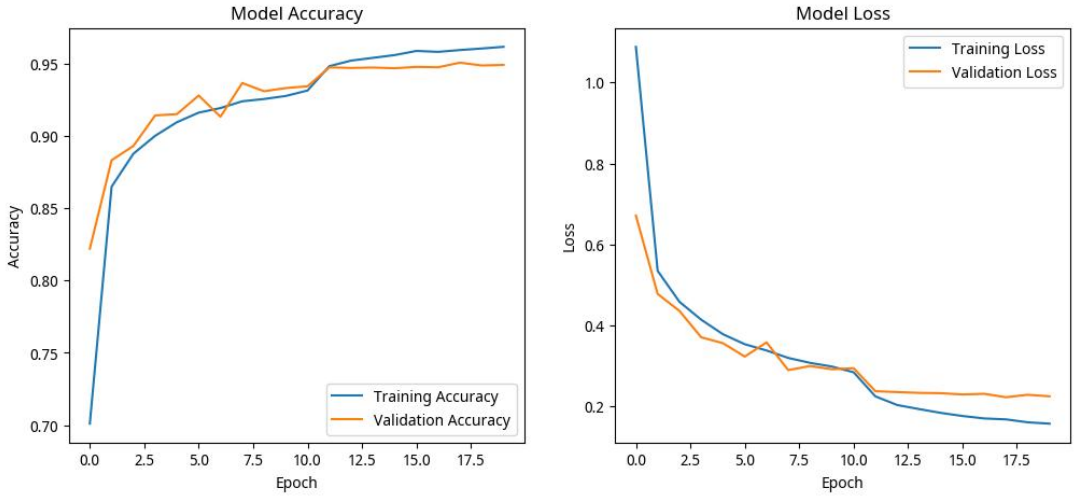

Figure 5.3: Accuracy on BanglaLekha-Isolated Dataset with consonants

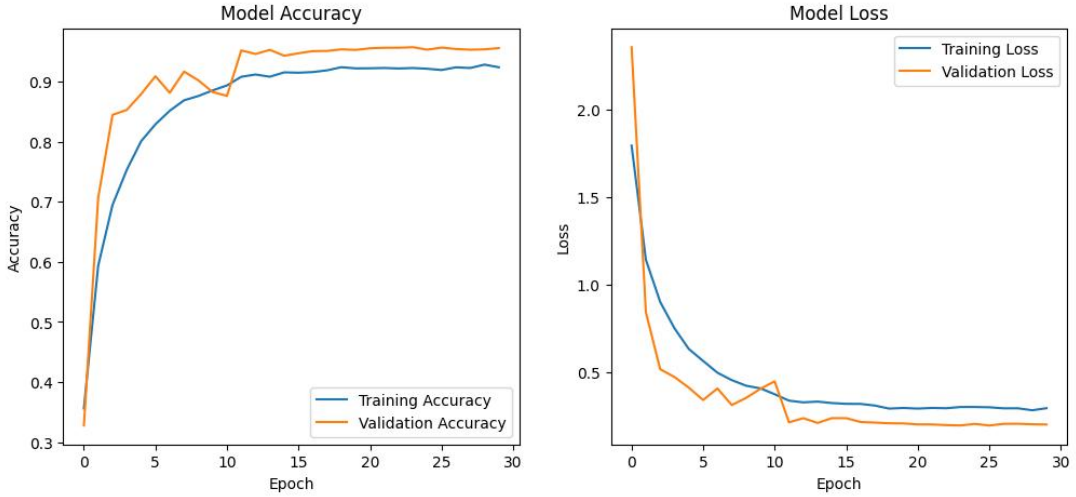

Figure 5.4: Accuracy on Ekush Dataset with digits

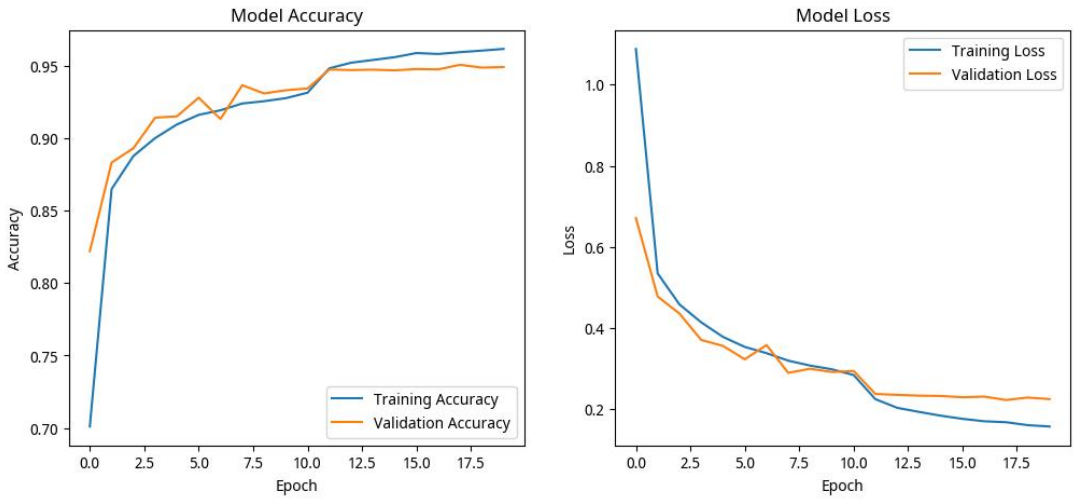

Figure 5.5: VGG16 Performance analysis on digits

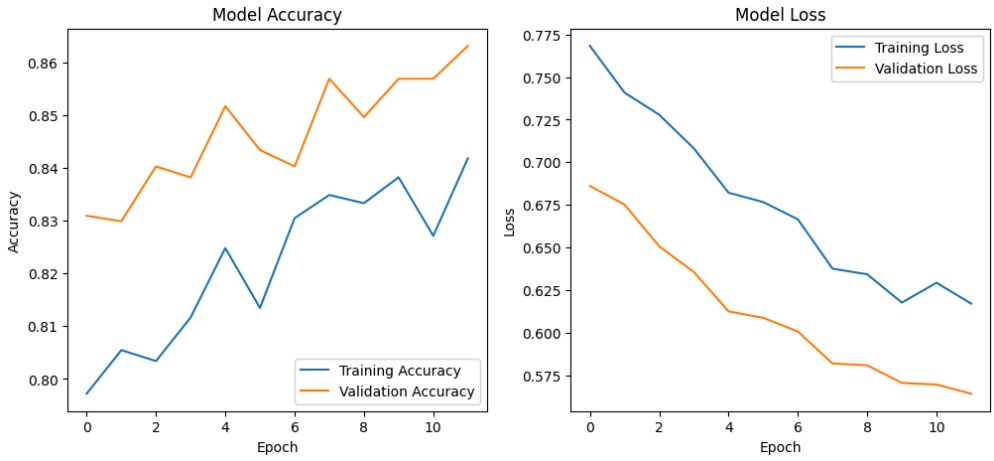

Figure 5.6: VGG16 Performance analysis on digits additional epochs

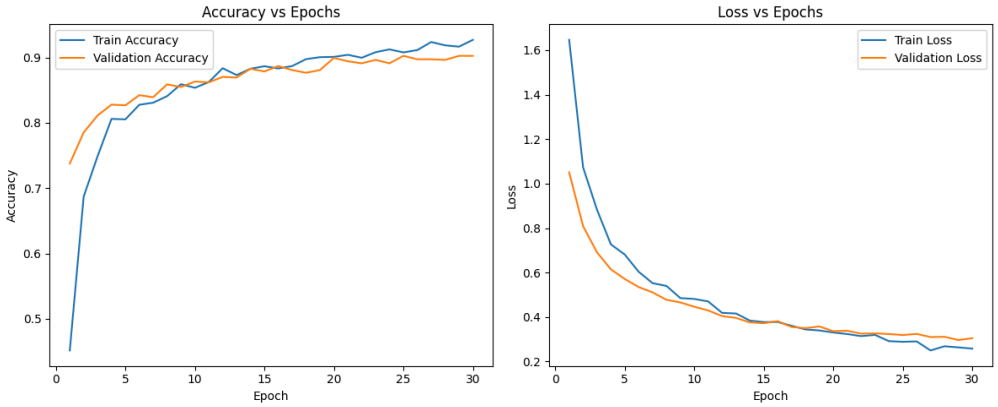

Figure 5.7: ResNet-50 performance analysis Accuracy Vs Loss

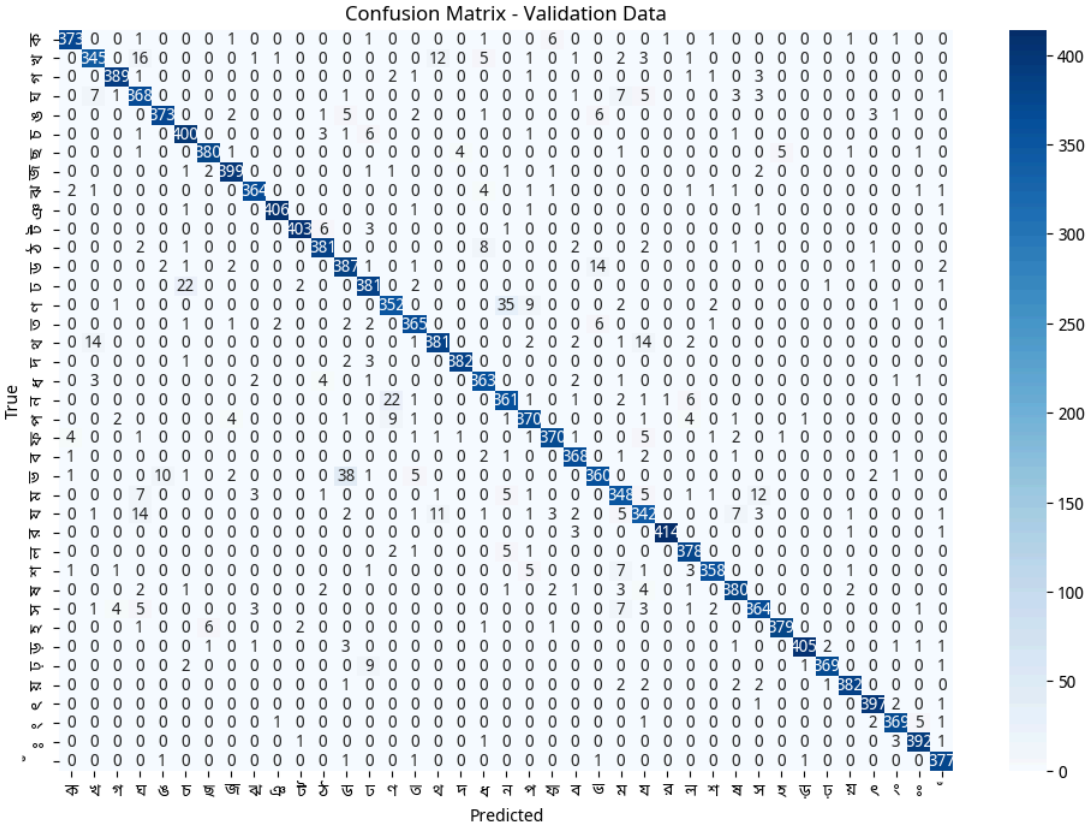

Figure 5.8: Confusion matrix on BanglaLekha Isolated dataset

Table 5.1: Performance Metrics on Ekush dataset

Character	precision	recall	f1-score	support
া	0.9615	0.9836	0.9724	305
ি	0.9903	0.9871	0.9887	309
ী	0.9733	0.9481	0.9605	308
ু	0.9272	0.9513	0.9391	308
ূ	0.9671	0.9608	0.9639	306

Continued on next page

Table 5.1 – Continued

Character	precision	recall	f1-score	support
ৃ	0.9578	0.9866	0.9720	299
ে	0.9797	0.9353	0.9570	309
ৈ	0.9832	0.9513	0.9670	308
ো	0.9709	0.9740	0.9724	308
ৌ	0.9738	0.9643	0.9690	308
অ	0.9801	0.9609	0.9704	307
আ	0.9803	0.9708	0.9755	308
ই	0.9868	0.9740	0.9804	308
ঈ	0.9771	0.9708	0.9739	308
উ	0.9195	0.9552	0.9370	335
ঊ	0.9846	0.9110	0.9464	281
ঋ	0.9801	0.9578	0.9688	308
এ	0.9899	0.9578	0.9736	308
ঐ	0.9837	0.9741	0.9789	309
ও	0.9052	0.9579	0.9308	309
ঔ	0.9899	0.9545	0.9719	308
ক	0.9732	0.9448	0.9588	308
খ	0.9373	0.9221	0.9296	308
গ	0.9744	0.9903	0.9823	308
ঘ	0.9418	0.8987	0.9197	306
ঙ	0.9453	0.9545	0.9499	308
চ	0.9644	0.9675	0.9660	308
ছ	0.9868	0.9707	0.9787	307
জ	0.9617	0.9805	0.9710	307
ঝ	0.9740	0.9740	0.9740	308
ঞ	0.9702	0.9452	0.9575	310
ট	0.9427	0.9642	0.9533	307
ঠ	0.9507	0.9444	0.9475	306
ড	0.8554	0.9281	0.8903	306
ঢ	0.9519	0.9738	0.9627	305
ণ	0.9525	0.9274	0.9398	303
ত	0.9522	0.9029	0.9269	309
থ	0.9330	0.9582	0.9455	407
দ	0.9381	0.9743	0.9558	311
ধ	0.9579	0.9610	0.9595	308
ন	0.9169	0.9644	0.9401	309
প	0.9680	0.8889	0.9267	306
ফ	0.9589	0.9838	0.9712	308
ব	0.9771	0.9708	0.9739	308
ভ	0.9213	0.9123	0.9168	308
ম	0.9138	0.9612	0.9369	309
য	0.9418	0.8721	0.9056	297
র	0.9778	0.9968	0.9872	309
ল	0.9834	0.9673	0.9753	306
শ	0.9767	0.9545	0.9655	308

Continued on next page

Table 5.1 – Continued

Character	precision	recall	f1-score	support
ষ	0.9219	0.9685	0.9446	317
স	0.9666	0.9444	0.9554	306
হ	0.9399	0.9706	0.9550	306
ড়	0.9901	0.9740	0.9820	308
ঢ়	0.9715	0.9935	0.9824	309
য়	0.9717	0.9810	0.9763	315
ৎ	0.9770	0.9739	0.9755	306
ং	0.9935	0.9870	0.9902	308
ঃ	0.9871	0.9935	0.9903	308
ঁ	0.9934	0.9771	0.9852	306
ব্দ	0.9772	0.9868	0.9820	304
ঙ্গ	0.9040	0.9605	0.9314	304
স্ক	0.9467	0.9251	0.9357	307
স্ফ	0.9316	0.9408	0.9362	304
চ্ছ	0.9610	0.9737	0.9673	304
স্থ	0.9759	0.9309	0.9529	304
ক্ত	0.8846	0.9803	0.9300	305
ম্ম	0.9061	0.9739	0.9388	307
ষ্ণ	0.9681	0.9870	0.9774	307
ম্প	0.8562	0.8867	0.8712	309
গু	0.9534	0.8303	0.8876	271
ষ্ব	0.9544	0.9379	0.9461	290
থ	0.9378	0.8660	0.9005	209
ড্ড	0.9720	0.9601	0.9660	326
ষ্ঠ	0.9612	0.9706	0.9659	306
ল্ল	0.9671	0.9515	0.9592	309
স্প	0.9133	0.8063	0.8565	222
ন্দ	0.9643	0.9674	0.9659	307
ন্ধ	0.9703	0.9577	0.9639	307
স্ম	0.8841	0.9416	0.9119	308
র্থ	0.9642	0.9642	0.9642	307
স্ত	0.8847	0.9594	0.9205	320
ষ্ট	0.9609	0.9609	0.9609	307
ন্ম	0.9500	0.9500	0.9500	300
ত	0.9000	0.8408	0.8694	289
জ্ঞ	0.9806	0.9870	0.9838	307
ন্ত্র	0.9964	0.9858	0.9911	281
ভ	0.9727	0.9283	0.9500	307
জ্ঞ	0.9037	0.8889	0.8962	306
ড্ড	0.9575	0.9544	0.9560	307
ক্ষ	0.9638	0.9513	0.9575	308
দ্ব	0.9785	0.8951	0.9349	305
চ্চ	0.9607	0.9575	0.9591	306
ক্র	0.9555	0.9555	0.9555	292
দ্দ	0.9901	0.9771	0.9836	306

Continued on next page

Table 5.1 – Continued

Character	precision	recall	f1-score	support
জ্জ	0.9375	0.9804	0.9585	306
ক্ক	0.9375	0.9836	0.9600	305
ন্ত	0.8591	0.8873	0.8730	213
ষ্ট	0.9672	0.9672	0.9672	305
ন্চ	0.9831	0.9479	0.9652	307
ট্ট	0.9525	0.9869	0.9694	305
শ্চ	0.9712	0.9935	0.9822	306
ক্ষ্ম	0.9869	0.9837	0.9853	307
জ্ব	0.9865	0.9511	0.9685	307
ঞ্চ	0.9369	0.9674	0.9519	307
দ্ধ	0.9009	0.9740	0.9360	308
ন্ন	0.9867	0.9770	0.9818	304
ঘ্ন	0.9847	0.9049	0.9431	284
ক্ল	0.9867	0.9738	0.9802	305
হ্ন	0.9494	0.9804	0.9646	306
ল্ত	0.9307	0.9307	0.9307	101
স্প	0.5493	0.7647	0.6393	51
০	0.9902	0.9870	0.9886	307
১	0.9839	0.9967	0.9903	307
২	0.9669	0.9542	0.9605	306
৩	0.9066	0.9773	0.9406	308
৪	0.9558	0.9870	0.9712	307
৫	0.9789	0.9055	0.9408	307
৬	0.9647	0.9805	0.9725	307
৭	0.9156	0.9186	0.9171	307
৮	0.9683	0.9935	0.9807	307
৯	0.9863	0.9379	0.9615	306

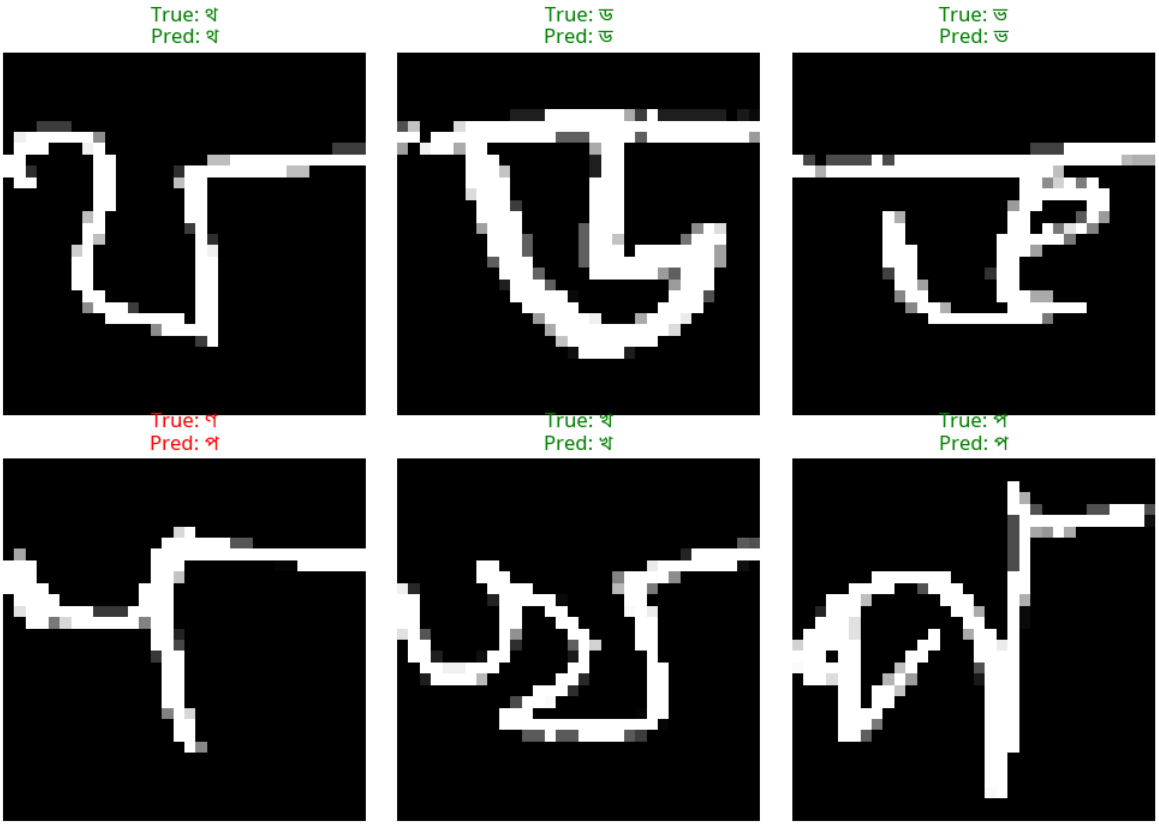

Figure 5.9: Prediction

Model	Accuracy	Val Accuracy	Loss	Val Loss	Dataset
Proposed CNN	94.29%	95.63%	25.35%	21.62%	Ekush(28x28)
Proposed CNN	96.52%	96.20%	17.25%	22.43%	BanglaLekha(28X28)
Proposed CNN	97.28%	97.89%	12.60%	11.23%	BanglaLekhaDigits(32X32)
Proposed CNN	92.71%	95.40%	28.58%	20.58%	Ekush digits(28x28)
ResNet 50	92.70%	90.25%	25.81%	30.47%	Ekush digits(224x224)
Vgg16	85.72%	86.31%	58.76%	56.42%	Ekush digits(224X224)

Table 5.2: Comparison Table on training and test accuracy, loss

From the above 5.2 we can see that our custom CNN model achieve 94.29% accuracy on Ekush Dataset with 28X28 size image. For BanglaLekha Dataset, 96.52% accuracy achieve using 32X32 size image. If we increase the image size to 32X32 pixel on BanglaLekha dataset (digits) accuracy get increased to 97.28% comparing with Ekush dataset (digits) with same model. On the contrary, ResNet-50 and VGG16 on Ekush dataset (digits) has lower accuracy than our proposed model. VGG16 does not perform mush well on the dataset. Comparing the models we can say that if we used better dataset we will achieve a good results with less computation.

5.1.1 Experimental Result

The proposed CNN model achieved an accuracy of 95.63% on the Ekush dataset, which has support for more compound characters. Finally, we deploy our pre-trained CNN model in our application and successfully recognize Bangla handwritten character.

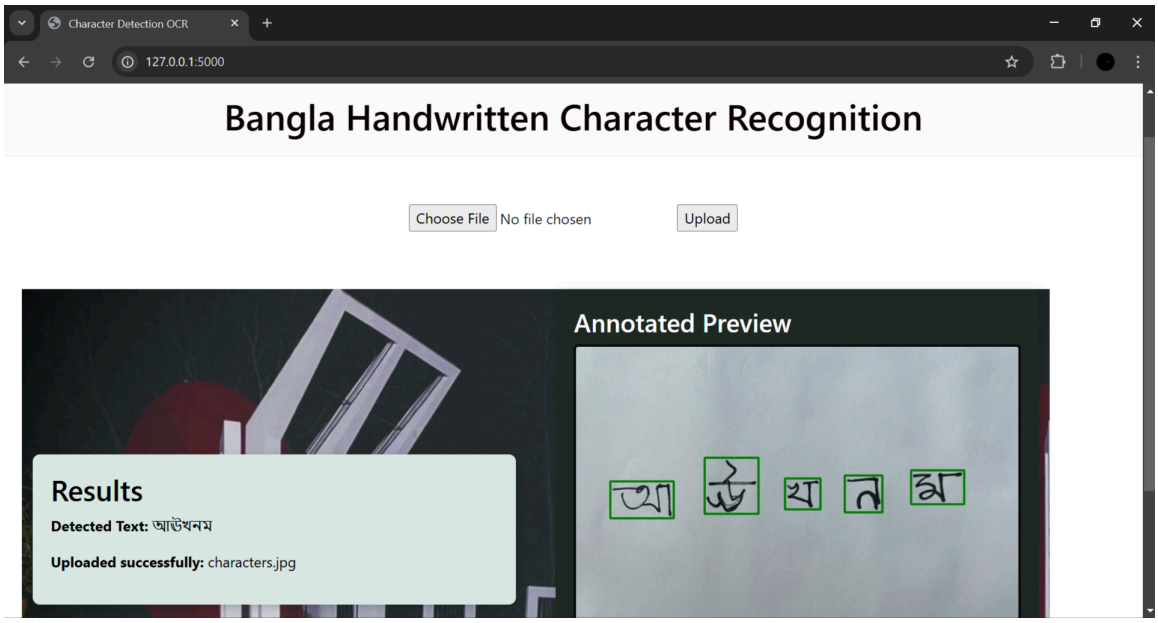

Figure 5.10: Character recognition using web application

Chapter 6

Conclusion

Bangla handwritten character recognition is one of the most difficult task due to its diversity. The research explored various deep learning approach to improve Bangla OCR. By implementing our model we got very promising results with 96.52%accuracy on BnaglaLekha-Isolated dataset with 32X32 pixel size images and 95.63% on Ekush Dataset with 28x28 pixel images. Ekush dataset has more support for compound characters, that's why we used it in our web application. Also, we found that if we increase the image size, the model performance gets better. In this model we used residual block to flow data more dynamically rather than use conventional CNN model. We simply add residual blocks to ensure the flow of data doesn't loss much information for during back propagation. Our main limitation is the scarcity of Bangla Dataset with good quality images with compound characters support. Also, due to unavailability of GPU, we could not stretch our model further for optimal result.

6.1 Future Works

In future, we aim to build a more robust model with less computational time. One of the main challenges in Bangla OCR is the lack of data diversity. Researchers should build more diverse dataset for Bangla language. More research should be conduct on dataset for better accuracy and detection and also should have support for frequent word list to train the model to learn better. Lack of quality dataset represents a alarming situation on this domain and affect the development of Bangla OCR for real world scenarios. Bangla Language has more than 300 compound characters. We can aim to build more diverse Bangla handwritten dataset with most possible compound characters with help of machine learning technique. For implementation in web application will focus on handling noise reduction in the preprocessing step and with real time user feedback we can adjust the model performance more robustly. The character recognition metrics shows that the application has potential in real world deployment.Further study needs to detect Bangla word and recognize using deep learning technique. The project provide a strong foundation on Bangla OCR for further development on this domain.

6.2 Conclusion

In conclusion, Bangla is one of most diverse and complex language with vowels, consonants, vowel modifiers, consonant modifiers. Also, the meaning and pronunciation changes due to adding those modifiers in the character. Furthermore, the diverse handwritten styles make it more difficult of Bangla OCR. This domain is in a blooming phase, further extensive research should be done for the betterment of Bangla OCR. This project contributes to the advancement of Bangla OCR by deploying deep learning techniques and a real world web application for character recognition. The findings of this study can be further extended to detect handwritten words and enhancing text recognition. To conclude, the analysis provides here offers a foundation on Bangla OCR and establishes a standard for future research.

Bibliography

[1] N. Otsu *et al.*, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[2] J. Serra, *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

[3] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[4] A. P. Witkin, "Scale-space filtering," in *Readings in computer vision*, Elsevier, 1987, pp. 329–332.

[5] B. Chaudhuri and U. Pal, "An ocr system to read two indian language scripts: Bangla and devnagari (hindi)," in *Proceedings of the fourth international conference on document analysis and recognition*, IEEE, vol. 2, 1997, pp. 1011–1015.

[6] B. B. Chaudhuri and U. Pal, "A complete printed bangla ocr system," *Pattern recognition*, vol. 31, no. 5, pp. 531–549, 1998.

[7] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[8] M. A. Hasnat, M. R. Chowdhury, and M. Khan, "An open source tesseract based optical character recognizer for bangla script," in *2009 10th international conference on document analysis and recognition*, IEEE, 2009, pp. 671–675.

[9] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, "High-performance ocr for printed english and fraktur using lstm networks," in *2013 12th international conference on document analysis and recognition*, IEEE, 2013, pp. 683–687.

[10] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive bengali word recognition using cnns with a recurrent model," in *2016 15th International conference on frontiers in handwriting recognition (ICFHR)*, IEEE, 2016, pp. 429–434.

[11] R. Azim, W. Rahman, and M. F. Karim, "Bangla hand-written character recognition using support vector machine," *International Journal of Engineering Works*, vol. 3, no. 6, pp. 36–46, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] M. A. R. Alif, S. Ahmed, and M. A. Hasan, "Isolated bangla handwritten character recognition with convolutional neural network," in *2017 20th International conference of computer and information technology (ICCIT)*, IEEE, 2017, pp. 1–6.

[14] A. Ashiquzzaman, A. K. Tushar, S. Dutta, and F. Mohsin, "An efficient method for improving classification accuracy of handwritten bangla compound characters using dcnn with dropout and elu," in *2017 Third international conference on research in computational intelligence and communication networks (ICRCICN)*, IEEE, 2017, pp. 147–152.

[15] M. Biswas, R. Islam, G. K. Shom, *et al.*, "Banglalekha-isolated: A multipurpose comprehensive dataset of handwritten bangla isolated characters," *Data in brief*, vol. 12, pp. 103–107, 2017.

[16] S. Sharif and M. Mahboob, "A comparison between hybrid models for classifying bangla isolated basic characters," in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, IEEE, 2017, pp. 211–216.

[17] A. S. A. Rabby, S. Haque, S. Islam, S. Abujar, and S. A. Hossain, "Bornonet: Bangla handwritten characters recognition using convolutional neural network," *Procedia computer science*, vol. 143, pp. 528–535, 2018.

[18] A. S. A. Rabby, S. Haque, S. A. Shahinoor, S. Abujar, and S. A. Hossain, "A universal way to collect and process handwritten data for any language," *Procedia computer science*, vol. 143, pp. 502–509, 2018.

[19] A. S. A. Rabby, S. Haque, M. S. Islam, S. Abujar, and S. A. Hossain, "Ekush: A multipurpose and multitype comprehensive database for online off-line bangla handwritten characters," in *Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018, Solapur, India, December 21–22, 2018, Revised Selected Papers, Part III 2*, Springer, 2019, pp. 149–158.

[20] G. A. Robby, A. Tandra, I. Susanto, J. Harefa, and A. Chowanda, "Implementation of optical character recognition using tesseract with the javanese script target in android application," *Procedia Computer Science*, vol. 157, pp. 499–505, 2019.

[21] M. A. Azad, H. S. Singha, and M. M. H. Nahid, "Bangla handwritten character recognition using deep convolutional autoencoder neural network," in *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, IEEE, 2020, pp. 295–300.

[22] A. Bennetot, I. Donadello, A. E. Qadi, *et al.*, "A practical guide on explainable ai techniques applied on biomedical use case applications," *arXiv preprint arXiv:2111.14260*, 2021.

[23] N. M. Dipu, S. A. Shohan, and K. Salam, "Bangla optical character recognition (ocr) using deep learning based image classification algorithms," in *2021 24th International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2021, pp. 1–5.

[24] T. Ghosh, M.-H.-Z. Abedin, H. Al Banna, N. Mumenin, and M. Abu Yousuf, "Performance analysis of state of the art convolutional neural network architectures in bangla handwritten character recognition," *Pattern Recognition and Image Analysis*, vol. 31, pp. 60–71, 2021.

[25] A. Y. Srizon and M. A. Hossain, "Bornomalanet: An effective approach for handwritten bengali characters recognition by using a low-cost convolutional neural network," in *2022 4th International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, IEEE, 2022, pp. 1–4.

[26] M. Asraful, M. A. Hossain, and E. Hossen, "Handwritten bengali alphabets, compound characters and numerals recognition using cnn-based approach," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 7, no. 3, pp. 60–77, 2023.