Bengali handwritten character recognition using convolutional neural network from live and real-world images

Project Report

submitted in the fulfilment of the requirements for the degree

Bachelor of Technology

in

Electronics and Communication Engineering

of

Maulana Abul Kalam Azad University of Technology

by

Aniket Bhunia

aniketbhunia007@gmail.com

ECE (2015-2019), Roll No-10900315013

Netaji Subhash Engineering College

&

Bishwajeet Ghosh

bishwajeetg01@gmail.com

ECE (2015-2019), Roll No-10900315030

Netaji Subhash Engineering College

Under the guidance of

Chira Ranjan Datta, Head of the Department

and

Dip Kumar Saha, Assistant Professor

Department of Electronics And Communication Engineering

Netaji Subhash Engineering College Techno City, Garia, Kolkata – 700 152

Certificate

This is to certify that this project report titled

Bengali Handwritten Character Recognition Using Convolutional Neural Network from Live and Real-World Images

submitted in fulfilment of requirements for award of the degree Bachelor of Technology (B. Tech) in E.C.E. of West Bengal University of Technology is a faithful record of the original work carried out by,

ANIKET BHUNIA, Roll no. 10900315013, Reg. No.151090110225

BISHWAJEET GHOSH, Roll no. 10900315030, Reg. No. 151090110242

under my guidance and supervision. It is further certified that it contains no material, which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except the assistances drawn from other sources, for which due acknowledgement has been made.

Date: ___/___/_____ _____

Guide's signature

_____

Head of the Department Electronics and Communication Engineering

Netaji Subhash Engineering College Techno City, Garia, Kolkata – 700 152

Declaration

We hereby declare that this project report titled

Bengali Handwritten Character Recognition Using Convolutional Neural Network from Live and Real-World Images

is our own original work carried out as a under-graduate student in Netaji Subhash Engineering College except to the extent that assistances from other sources are duly acknowledged.

All sources used for this project report have been fully and properly cited. It contains no material which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except where due acknowledgement is made.

Student's names	Signatures	Dates
Aniket Bhunia	_____	___/___/_____
Bishwajeet Ghosh	_____	___/___/_____

Certificate of approval

We hereby approve this dissertation titled

Bengali Handwritten Character Recognition Using Convolutional Neural Network from Live and Real-World Images

carried out by

ANIKET BHUNIA, Roll no. 10900315013, Regd. No. 151090110225

BISHWAJEET GHOSH, Roll no. 10900315030, Regd. No. 151090110242

under the guidance of

CHIRA RANJAN DATTA

DIP KUMAR SAHA

of Netaji Subhash Engineering College, Kolkata in partial fulfilment of requirements for award of

the degree Bachelor of Technology (B. Tech) in Electronics and Communication Engineering of

Maulana Abdul Kalam Azad University of Technology

Date:___/___/_____

Examiners' signatures:

1. ……

2. ……

3. ……

Acknowledgement

We would like to express our special thanks of gratitude to our professor Dip Kumar Saha as well as our Head of the Department Chira Ranjan Datta who gave us the golden opportunity to do this wonderful project on the topic Live Hand-written digit recognition using multiple Machine Learning Algorithms which also helped me in doing a lot of Research and we came to know about so many new things we are really thankful to them. Secondly, we would also like to thank our teachers and friends who helped us a lot in finalizing this project within the limited time frame.

Aniket Bhunia

Bishwajeet Ghosh

Dated……

Abstract

This project seeks to identify individual handwritten Bengali character. We used three different approaches to take inputs. A light weight convolutional neural network model was trained, validated and tested with CMATERdb dataset, developed by Jadavpur University, Kolkata. We used two main approaches to accomplish the identification task, classifying the character directly and character segmentation.

Handwritten characters recognition has always a big challenge due to its variation of shape, size, and writing style. Accurate handwritten recognition is becoming more thoughtful to the researchers for its educational and economic values. There had several works been already done on the Bangla Handwritten Character Recognition, but still there is no robust model developed yet. Therefore, this paper states development and implementation of a lightweight CNN model for classifying Bangla Handwriting characters. The proposed model can take any type of input image which may be live, may be pre-clicked or maybe written in canvas which we built from scratch and classify that.

We have used python programming language(3.6) and libraries such as 'keras' (to build CNN), 'openCV' for image processing, 'flask' (to build canvas), 'tensorflow' as backend for training and 'theano' as backend in testing. The whole project can be found on this link https://github.com/aniketbhunia007/fyp

Keywords: Bangla handwritten character recognition, Convolutional neural network, Pattern recognition, Deep learning, Computer vision, Machine learning, Python, Keras, OpenCV, Image processing

Contents

Introduction

Handwritten character recognition development research is rapidly evolving and reshaping the must automation fields like—automatic check reading, automatic number plate reading, digital postal service, Optical Image Recognizing (OCR), etc. Due to its various aspects of uses, computer vision researchers verily feel to work on it and improve quality and performance indeed. But handwritten recognizing is more challenging compared to the typed letter. Because different people write in a different way and which creates a higher degree of variance in written style. Also, there are some similarities between different characters shape. The situation of overwriting makes it more challenging for accurately classifying the handwritten digit.

Nowadays, deep learning, especially the Convolutional neural network (CNN) is working better in the purpose of classifying these types of recognition work rather other machine learning methods like SVM or Support Vector Machines, KNN or K-nearest Neighbours. In our first part of the project we learned about SVM and KNN and how CNN is more efficient and in this final part we implemented our model to do the recognition task.

Bengali, also known by its endonym Bangla, is an Indo-Aryan language primarily spoken by the Bengalis in South Asia. It is the official and most widely spoken language of Bangladesh and second most widely spoken of the 22 scheduled languages of India, behind Hindi. It is the 7th most spoken language in the world. Bengali script has 50 letters including 11 vowels and 39 consonants.

We approach this recognition problem with character images because CNN stand to work better on raw input pixels rather than features or parts of an image. In both of our techniques, our models take in an image of a character and output the name of the character as well as the accuracy. We have achieved 89.26 % testing accuracy with our CNN model.

Literature Review

OCR, Feature extraction and related work

OCR consists of many phases such as Scanning of image, Pre-processing, Segmentation, Feature Extraction, Classifications and Recognition, Post Processing. The task of pre-processing relates to the removal of noise and variation in the image [1]. In scanning step, the image is acquired. The quality of image depends highly on the scanner being sed. In practical applications, the scanned images are not perfect there may be some noise due to some unnecessary details in the image which can cause a disruption in the detection of the characters in the image.[19] Pre-processing involves removal of noise (applying filters like Gaussian filter, Gabor filter etc.) and proper conversion of image like a coloured image can be converted into grayscale or binary image for further processing of image.

Feature extraction involves recognizing the feature required.[20] One of the most important phases in successfully achieving character recognition is the task of feature extraction. Feature extraction stage identifies and extracts various attributes from characters that help distinctly and uniquely distinguish different characters. A number of different feature extraction methods have been proposed in literature in accordance with different character representations. For example, different sets of features have been defined to best represent character shapes, boundaries, their skeletons and strokes etc. Trier et al. [2] comprehensively describe different types of features and methods for character recognition task. Among these methods, there are statistical feature extractors and structural feature extractors. Statistical features consider the statistical distribution of pixel values. Major statistical features used for handwritten character recognition task include zoning, projections, profiles, and crossings etc. Structural features consider the geometry and topology of character samples such as number of loops, end points, junction points, aspect ratio, type of strokes and their directions etc. Some feature extraction methods are based on different transformations such as those based on Fourier transform, wavelet transform, central moments, and Zernike moments etc. In [3], the authors describe a zoning-based feature extractor to recognize handwritten numerals of Indian Kannada script. Authors in [4] recognize handwritten numerals using Fourier descriptors and neural network. In [5], the authors recognize Chinese

handwritten characters using gradient and wavelet-based features. In [6], the authors extract moment-based features in order to recognize handwritten Arabic letters. They use genetic algorithm for feature selection and use SVM to evaluate the classification error for the chosen feature subset.

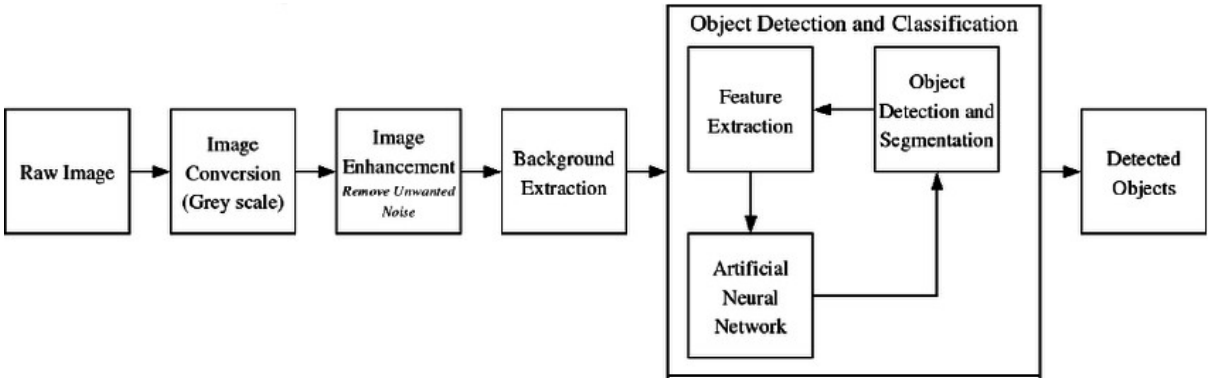


Fig : Feature Extraction Steps in Image Processing

Instead of focusing on feature vector based on a single representation of a character, it is a trend now of combining different types of features extracted from different representations of the same character. The advantage of combining, and harnessing, such different kinds of features is that it can offer wider range of identification clues to help improve the accuracy of recognition. For example, Hettes et al. [7] combine different statistical and structural features for recognition of handwritten characters. They construct a 124-variable feature vector comprising following seven families of features: 1) intersection of the character with horizontal and vertical straight lines, 2) invariant moments, 3) holes and concave arcs, 4) extremes, 5) end points and junction points 6) profiles, and 7) projections. Aurora et al. [8] combine different feature extraction techniques such as intersection-based features, shadow features, chain code and curve fitting features for Indian Devanagari language script. Kimura et al. [9] propose a genetic algorithm-based strategy for finding a suitable combination of features from a large pool of features with the objective criteria to minimize the classification error.

Pattern classification, SVM and related work

The second most important component in successfully achieving handwritten character recognition is the pattern classification stage. This stage will assign an unknown character sample to one of possible classes by utilizing the information of feature extraction stage. Different types of classifiers can be

built based on the nature and type of data samples and the extracted features. [26]

Classifiers used for character recognition problem include k-nearest neighbour classifier, hidden Markov model (HMM), support vector machine (SVM), and artificial neural network (ANN) etc. Jain et al. [10] give a review of statistical pattern recognition techniques. In [11], Pal and Singh train neural network to recognize uppercase handwritten characters based on Fourier descriptors of character boundaries as features. In [12], recognition of handwritten alphabets using neural network and zoning based diagonal features is addressed. In [13], Shubhangi and Hiremath recognize English handwritten characters and digits by extracting structural micro features for SVM classifier. Nasien et al. [14] also use SVM classifier to recognize handwritten alphabets by employing Freeman Chain codes as the features. In [15], Train et al. recognize accented handwritten French characters based on a combination of structural and moment features for SVM classifier. In [16], Liu and Nakagawa give a review of learning methods for nearest neighbour classifiers. [17] and [18] build HMM to recognize, respectively, offline handwritten Chinese characters and online English characters.

Bengali Hand-written Character Recognition and Related work

Analysis of the structural features of the letterforms is often successfully used in handwritten character recognition.

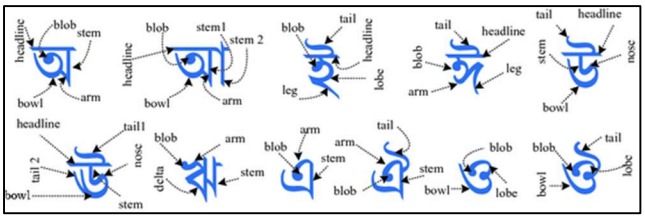


Fig: Bengali Character Writing Style

In 2006, Chowdhury et al. [20] developed a method for recognizing Bengali handwritten numerals where the characters are modelled as water reservoirs. In 2011, Mandal [21] developed a Bengali handwritten character recognition scheme based on the analysis of gradient features. Recently, Das et al. developed a method for recognizing handwritten Bengali numerals using mathematical morphology [22].

The table below shows the Bengali characters in printed form along with the phonetics in English. The 50 basic characters can be further combined to create several complex alphabets with mixed sound but this is out of scope of the current project.[25]

Vowels:								
1	অ	aw	6	ঊ	uuu	11	ঔ	ou
2	আ	aaa	7	ঋ	rhi			
3	ই	e	8	এ	ey			
4	ঈ	eee	9	ঐ	oi			
5	উ	u	10	ও	o			

Consonants:								
12	ক	ka	17	চ	cha	22	ট	taw
13	খ	kha	18	ছ	chha	23	ঠ	thaw
14	গ	ga	19	জ	ja	24	ড	daw
15	ঘ	gha	20	ঝ	jha	25	ঢ	dhaw
16	ঙ	nga	21	ঞ	nya	26	ণ	naw(1)
27	ত	ta	32	প	paw	37	য	jaw
28	থ	tha	33	ফ	phaw	38	র	raw
29	দ	da	34	ব	baw	39	ল	law
30	ধ	dha	35	ভ	bhaw	40	শ	saw(1)
31	ন	naw(2)	36	ম	maw	41	ষ	saw(2)
42	স	saw(3)	47	ৎ	khando-taw			
43	হ	haw	48	◌ং	onussar			
44	ড়	dra	49	:	bisargo			

45	ঢ়	dhra	50	◌ঁ◌	chandrabindu
46	য়	ya			

Ray and Chatterjee [23] did the first significant work in Bengali HCR. After that, many more researchers tried several other methods for improving the performance of Bangla Handwritten Character Recognition (HCR) as evident in [24]. Hasnat et al. [32], proposed an HCR capable of classifying both printed and handwritten characters by applying Discrete Cosine Transform (DCT) over the input image and Hidden Markov Model (HMM) for character classification. Wen et al. [33] proposed a Bangla numerals recognition method using Principal Component Analysis (PCA) and Support Vector Machines. Liu and Suen [34], proposed a method of identifying both Farsi and Bangla Numerals. In Hassan and Khan [35], K-NN algorithm was used where features were extracted using local binary patterns. Das et al. [36], proposed a feature set representation for Bangla handwritten alphabets recognition which was a combination of 8 distance features, 24 shadow features, 84 quad trees based longest run features and 16 centroid features. Their accuracy was 85.40% on a 50-character class dataset. The above-mentioned methods however used many handcrafted features extracted for small dataset which turned out to be unsuitable for deploying solutions.[27]

CNN and Related Work

The CNN structure was first time proposed by Fukushima in 1980 [37]. However, it has not been widely used because the training algorithm was not easy to use. In 1990s, LeCun et al. applied a gradient-based learning algorithm to CNN and obtained successful results [38]. After that, researchers further improved CNN and reported good results in pattern recognition. Recently, Cirean et al. applied multi-column CNNs to recognize digits, alpha-numerals, traffic signs, and the other object class [39]. They reported excellent results and surpassed conventional best records on many benchmark databases, including MNIST [38] handwritten digits database and CIFAR-10 [40].

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)\,d\tau$$

Fig: Convolution

Why CNN?

CNNs can be thought of automatic feature extractors from the image. While if I use an algorithm with pixel vector, I lose a lot of spatial interaction between pixels, a CNN effectively uses adjacent pixel information to effectively down sample the image first by convolution and then uses a prediction layer at the end.[41]

Ruggedness to shifts and distortion in the image: Detection using CNN is rugged to distortions such as change in shape due to camera lens, different lighting conditions, different poses, presence of partial occlusions, horizontal and vertical shifts, etc. However, CNNs are shift invariant since the same weight configuration is used across space. In theory, we also can achieve shift invalidates using fully connected layers.[28] But the outcome of training in this case is multiple units with identical weight patterns at different locations of the input. To learn these weight configurations, a large number of training instances would be required to cover the space of possible variations.[46]

Fewer Memory requirements: In this same hypothetical case where we use a fully connected layer to extract the features, the input image of size 32x32 and a hidden layer having 1000 features will require an order of 106 coefficients, a huge memory requirement. In the convolutional layer, the same coefficients are used across different locations in the space, so the memory requirement is drastically reduced.[47]

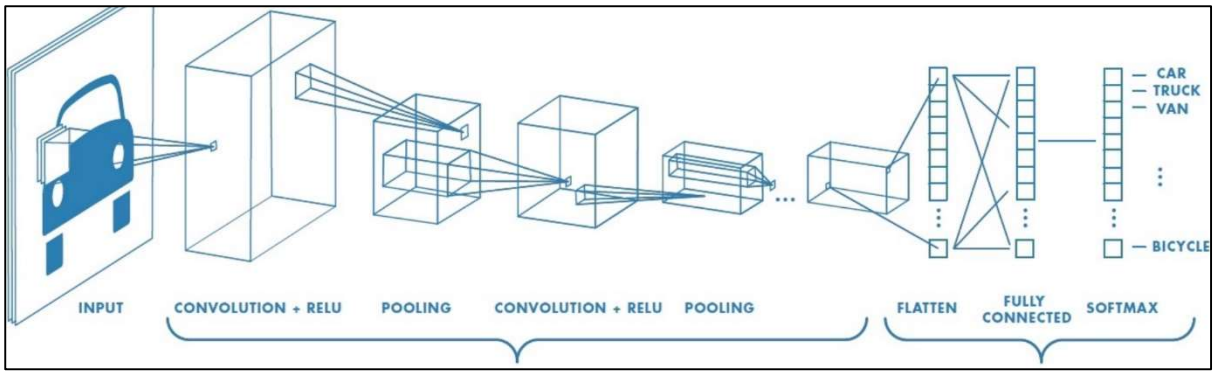
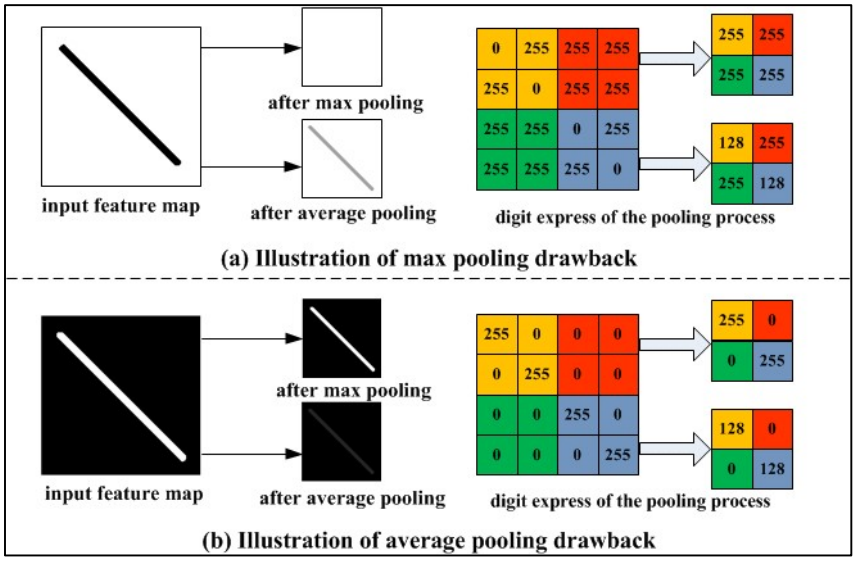


Fig : Basic CNN Layer Working Diagram

In the simple neural networks, each neuron was fully connected to each of the neurons in the subsequent layer. More concretely, each neuron in the hidden layer computed a function that depended on the values of every node in the input layer. In visual recognition, however, it is often advantageous to exploit local substructure within the image. [29] For example, pixels that are close together in the image (e.g., adjacent pixels) tend to be strongly correlated

while pixels that are far apart in the image tend to be weakly correlated or uncorrelated. Not surprisingly then, many standard feature representations used in computer vision problems are based upon local features within the image [30]. In the CNN architecture, we capture this local substructure within the image by constraining each neuron to depend only on a spatially local subset of the variables in the previous layer. For example, if the input to the CNN is a 32-by-32 image patch, a neuron in the first hidden layer might only depend on an 8-by-8 sub window within the overall 32-by-32 window. These of nodes in the input layer that affect the activation of a neuron is referred to as the neuron's receptive field. Intuitively, this is the part of the image that the neuron sees.

These high number of filters essentially learn to capture spatial features from the image based on the learned weights through back propagation and stacked layers of filters can be used to detect complex spatial shapes from the spatial features at every subsequent level. [57] The second feature that distinguishes CNNs from simple neural networks is the fact that the edge weights in the network are shared across different neurons in the hidden layers. Recall that each neuron in the network first computes a weighted linear combination of its inputs.[31] We can view this process as evaluating a linear filter over the input values. In this context,
sharing the weights across multiple neurons in a hidden layer translates to evaluating haemofilter over multiple sub windows of the input image. In this regard, we can view the CNN as effectively learning a set of filters each of which is applied to all of the sub windows within the input image. Using the same set of filters over the entire image forces the network to learn a general encoding or representation of the underlying data. [56] Constraining the weights to be



equal across different neurons also has a regularizing effect on the CNN; in turn, this allows the network to generalize bettering many visual recognition settings. Another benefit of weight sharing is the fact that it substantially

reduces the number of free parameters in the CNN, making it markedly easier and more efficient to train.[49] As a final note, evaluating a filter over each window in the input image amounts to performing a convolution of the image with the filter (deconvolved image with the filter). Thus, in the convolutional step of the CNN, we take the input image and convolve it with each filter into obtain the convolutional response map. The final distinguishing component in a CNN is the presence of subsampling or pooling layers. The goal here is twofold: reduce the dimensionality of the convolutional responses and confer a small degree of translational invariance into the model.

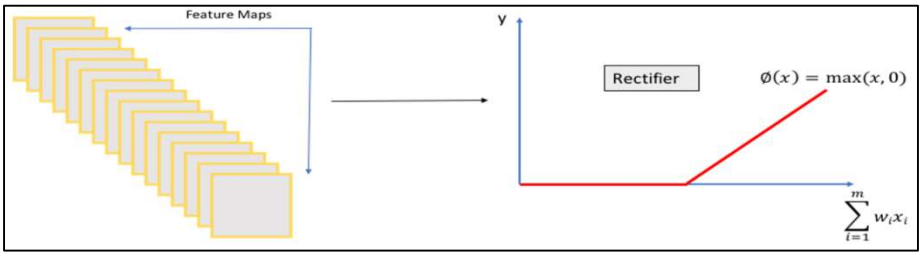


Fig: The ReLU rectifier to enforce non-linearity for sharp feature boundaries

We then evaluate a pooling function over the responses in each block. This process yields a smaller response map with dimension (one response for each block). In the case of max pooling, the response for each block taken to be the maximum value over the block responses, and in the case of average pooling, the response is taken to be the average value of the block responses. [51] An example of average pooling. In this case, the convolutional response map is a4-by-4 grid and we average pool over four 2-by-2 blocks arranged in a 2-by-2 grid. The pooled response is taken to be the average of the values in the block. After applying this average pooling procedure, we arrive at a final 2-by-2 pooled response map. Compared to the original 4-by-4 convolutional response map, this represents significant reduction in dimensionality of the response map. In a typical CNN, we have multiple layers, alternating between convolution and pooling. [52] For example, we can stack another convolution-pooling layer on top of the outputs of the first convolution-pooling layer. In this case, we simply treat the outputs of the first setoff convolution-pooling layers as the input to the second set of layers. In this way, we can construct a multi-layered or deep architecture. Intuitively, the low-level convolutional filters, such as those in the first convolutional layer, can be thought of as providing a low-level encoding of the input data. In the case of image data, these low-level filters may consist of simple edge filters.[56] As we move to higher layers in the neural network, the model begins to learn more and more complicated structures.

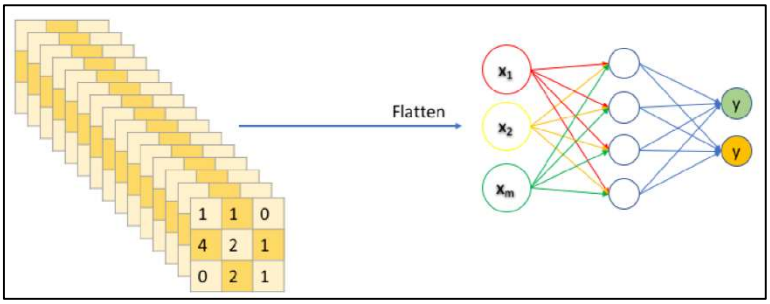


Fig: Illustration of Denselayers after the convolution operations.

By using multiple layers and large numbers of filters, the CNN architecture can thus provide vast amounts of representational power. To train a CNN, we can use the standard technique of error backpropagation used to train neural networks [53]. Convolutional neural networks have enjoyed a series of successes in many problems related to text classification such as handwriting recognition visual object recognition and character recognition Coupled with the rapid advancements in distributed and GPU (graphics processing units) computation, it is now possible to train much larger and more powerful CNNs that achieve state-of-the-art performance on standard benchmarks [54]. Thus, by leveraging the representational capacity contained within these networks in conjunction with the robustness of features derived from unsupervised algorithms, were able to construct simple, but powerful and robust, systems for both text detection and recognition.

Apart from there also present several Bangla Handwritten Character Recognition and had achieved pretty good success. Halima Begum et al, "Recognition of Handwritten Bangla Characters using Gabor Filter and Artificial Neural Network" [42] works with own dataset that was collected from 95 volunteers and their proposed model achieved without feature extraction and with feature extraction around 68.9% and 79:4% of recognition rate respectively. "Recognition of Handwritten Bangla Basic Character and Digit Using Convex Hull Basic Feature" [43] accuracy for Bangla character 76.86% and Bangla numeral 99.45%. "Bangla Handwritten Character Recognition using Convolutional Neural Network" [44] achieved 85.36% test accuracy using their own dataset. In "Handwritten Bangla Basic and Compound character recognition using MLP and SVM classifier" [45] handwritten Bangla character recognition with MLP and SVM has been proposed and they achieved around 79.73% and 80.9% of recognition rate, respectively. Using these robust and highly-accurate components renders it possible to obtain full end-to-end results using only the simplest of post-processing techniques.[55]

Methodology

.

Problem Statement

The objective of this project is to recognize Bengali character (both printed and hand-written) using CNN model where the user is giving input by clicking image, by writing in canvas and by pre-clicked image

Dataset

We are using CMATERdb 3.1.2 dataset is developed by Jadavpur University, Kolkata. It has 15000 images of character. The images were 32 x 32px noise-free images and the images edge look blocker. The data are split into 12000 train data and 3000 test data. The actual dataset that we use to train the model (weights and biases in the case of Neural Network). The model sees and learns from this data. The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

Fig: Image of 'aaa'

Data preprocessing

We have pre-processed the images to improve the quality of the CNN algorithm:

(1) Converting: We first translated all the data images to black and white (mode "L"), the library uses the ITU-R 601-2 luma transform:

```
1. L = R * 299/1000 + G * 587/1000 + B * 114/1000
```

(2) Rescaling: We rescale the images by dividing every pixel in every image by 255. So, the scale is now 0-1 instead of 0-255.
(3) We have randomly cropped the input images into 32x32 greyscale images.

(4) One-hot encoding for the categories to compare in the classification stage. Labels start from 0 to 49 for the 50 classes.

```
1. Before one hot encoding:  2
2. After one hot encoding: [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
```

(5) Splitting: Then we split the data into train and validation data. The sizes of train data are 9600 and validation data is 2400. The validation data is sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

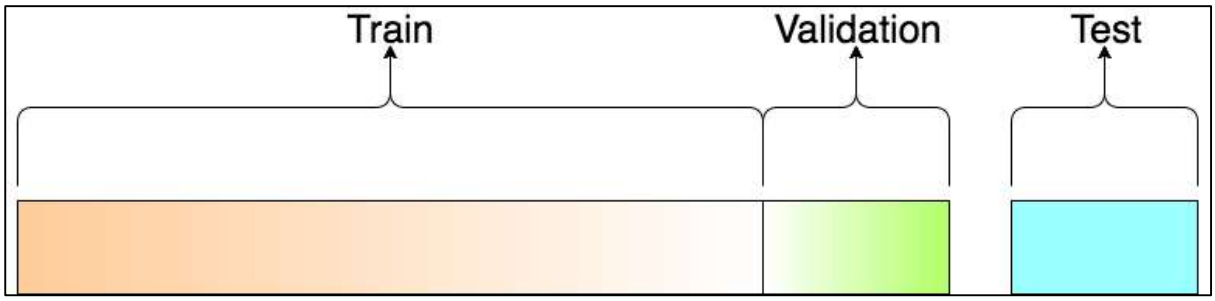


Fig: A visualisation of the splits

Model

The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the 'add()' function to add layers to our model. Our first layer is Conv2D layer. This convolution layer that will deal with our input images, which are seen as 2-dimensional matrices. 32 in the second layer is the number of nodes in layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In our case, 32 work well, so we will stick with this for now. Kernel size is the size of the filter matrix for our convolution. So, a kernel size of 3,3 means we will have a 3x3 filter matrix.

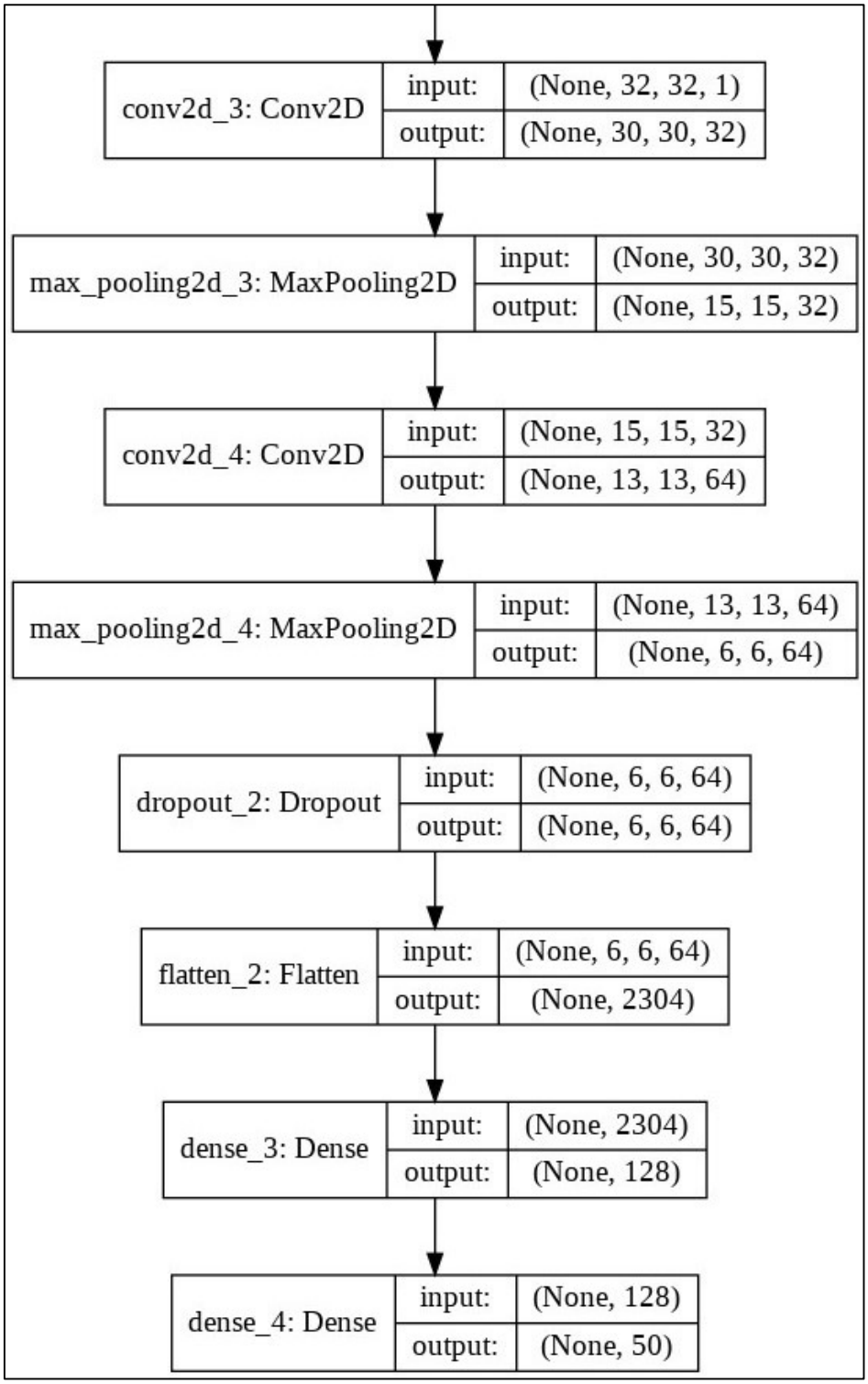
Activation is the activation function for the layer. The activation function we will be using for our is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks. Our first layer also takes in an input shape. This is the shape of each input image, 32,32,1 as seen earlier on, with the 1 signifying that the images are greyscale.

In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers. Dense' is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks. We will have 128 nodes in our output layer, one for each possible outcome (0–127).

The activation is 'softmax'. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability. We need to compile our model. Compiling the model takes three parameters: optimizer, loss and metrics.

The optimizer controls the learning rate. We will be using 'adam' as our optimizer. Adam is generally a good optimizer to use for many cases. Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter.

We will use 'categorical_crossentropy' for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better. To make things even easier to interpret, we will use the 'accuracy' metric to see the accuracy score on the validation set when we train the model. Then we will train our model.



Model Summary

Training and Testing Model

To train, we will use the 'fit ()' function on our model with the following parameters: training data (train_X), target data (train_y), validation data, and the number of epochs. The verbose flag, set to 1 here, specifies if you want detailed information being printed in the console about the progress of the training. For our validation data, we will use the test set provided to us in our dataset, which we have split into X_dev and Y_dev. The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. To test the model we used evaluate () function which is also a part of keras library.

Input Methods

There are three types of data input methods are written. They are:

1. (Method A) Live data input using device camera

2. (Method B) Pre-clicked image in different format i.e. '.bmp', '.dib', '.jpeg', '.jpg', '.jpe', '.jp2', '.png', '.webp', '.pbm', '.pgm', '.ppm', '.sr', '.ras', '.tiff', '.tif'

3. (Method C) Writing character in given canvas

Image segmentation

Image segmentation is a very important image processing step. It is an active area of research with applications ranging from computer vision to medical imagery to traffic and video surveillance. We have created two segmentation methods i.e. contour-based image segmentation and histogram-based image segmentation.

A histogram is a graph showing the number of pixels in an image at different intensity values found in that image. Simply put, a histogram is a graph wherein the x-axis shows all the values that are in the image while the y-axis shows the frequency of those values. In 'imscanH' function which we used in histogram-based segmentation we have done following steps. i.e. scanned image by histogram, read image, preprocessed image, scanned image along rows, scanned image along columns, drew boundaries on image, plotted histogram.

In contour-based image segmentation at first the markers and background are detected. These markers are pixels that we label unambiguously as either object which in this case is character or background. **Contour segmentation** also called as **snakes** and is initialized using automated contour or line, around the area of interest and this contour then slowly contracts and is attracted or repelled from light and edges. In 'imscanC' function which we used in contour-based segmentation we have done following steps. i.e. scanned image for contours, read image, preprocessed image, found contours, found bounding rectangle around each contour, sorted bounding rectangles from left to right, processed each bounding rectangle and attributes of bounding rectangle, ignored tiny objects assuming them as noise, drew bounding rectangle on image, extracted region of interest from threshold image using attributes of bounding rectangle.

For creating these segmentation functions, we have created different functions. i.e. imscan_rows, impreprocess, imread, endpoints, imscan_cols, imdraw_boundary, imdraw_bbox, plot_hist. In endpoints function we have created end-points of a 1D binary array. We have created imread function to read the image. In impreprocess function we have converted image to grayscale, then applied gaussian blurring to remove noise and then thresholded image which is explained as If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). In imscan_cols function we scanned image along columns, then detected number of lines, then initialized and populated accumulator and then found word segments along columns of each line segment. In imscan_rows functionwe scanned image along rows, then initialized and populated accumulator and then found line segments along rows. In imdraw_boundary function we drew boundaries on image with line boundaries and word boundaries. In imdraw_bbox function we drew bounding boxes on image.

Work Flow

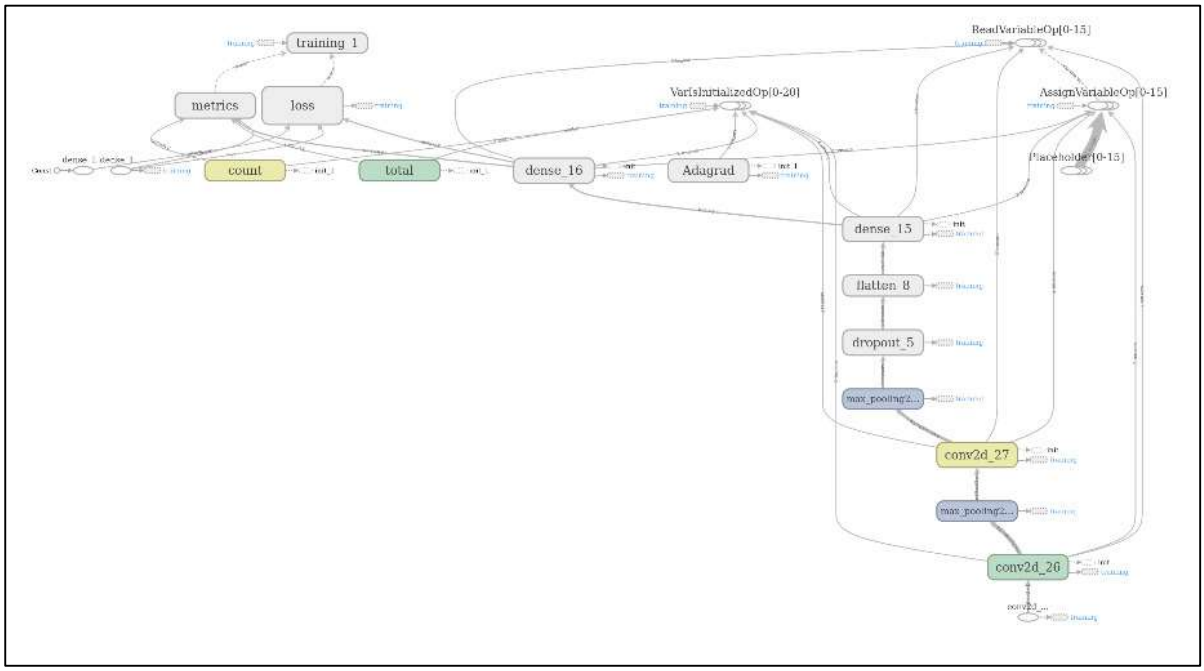


Fig- Diagram of how our CNN model works

Application Development

There are two applications are created. The first one is created with Flask library. It can take pre-clicked or written in canvas image to predict character.

Within flaskapp/, create a folder, app/, to contain all your files. Inside app/, create a folder *static/*; this is where we'll put our web app's images, CSS, and JavaScript files, so create folders for each of those, as demonstrated above. Additionally, create another folder, templates/, to store the app's web templates. Create an empty Python file routes.py for the application logic, such as URL routing.

```
1. ├── app
2. │   ├── static
3. │   │   ├── css
4. │   │   ├── img
5. │   │   └── js
6. │   ├── templates
7. │   └── routes.py
```

A user issues a request for a domain's root URL / to go to its home page, routes.py maps the URL / to a Python function, The Python function finds a web template living in the *t*emplates/ folder. A web template will look in the static/ folder for any images, CSS, or JavaScript files it needs as it renders to HTML. Rendered HTML is sent back to routes.py, routes.py sends the HTML back to the browser.

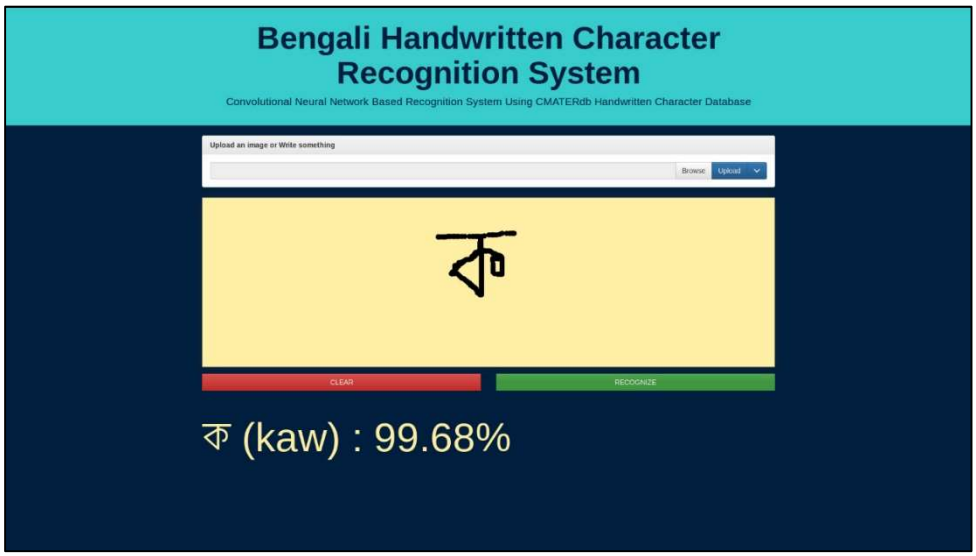


Fig: The frontend view of the application

We created a function name ocr () which scanned input image and found regions of interest used the two image segmentation methods we have discussed in methodology, processed each region of interest, resized and padded image, perform negation to produce a binary image similar to training images, reshaped and scaled features and predict the image in return. In app part of flask we created a recognize function which used the ocr function to return the result prediction. We start with a request issued from a web browser. A user types a URL into the address bar. The request hits routes.py, which has code that maps the URL to a function. The function finds a template in the *templates/* folder, renders it to HTML, and sends it back to the browser. That means when the user navigates to localhost:5000, the *home* function will run and it will return its output on the webpage.

In the second application the OpenCV library is used where the library simply access the camera using VideoCapture(0) operation and live image would be taken as input and converted into binary image then smoothed and resized and then the character will be predicted. We have to load the model which we have to save as .h5 file before all this operation.

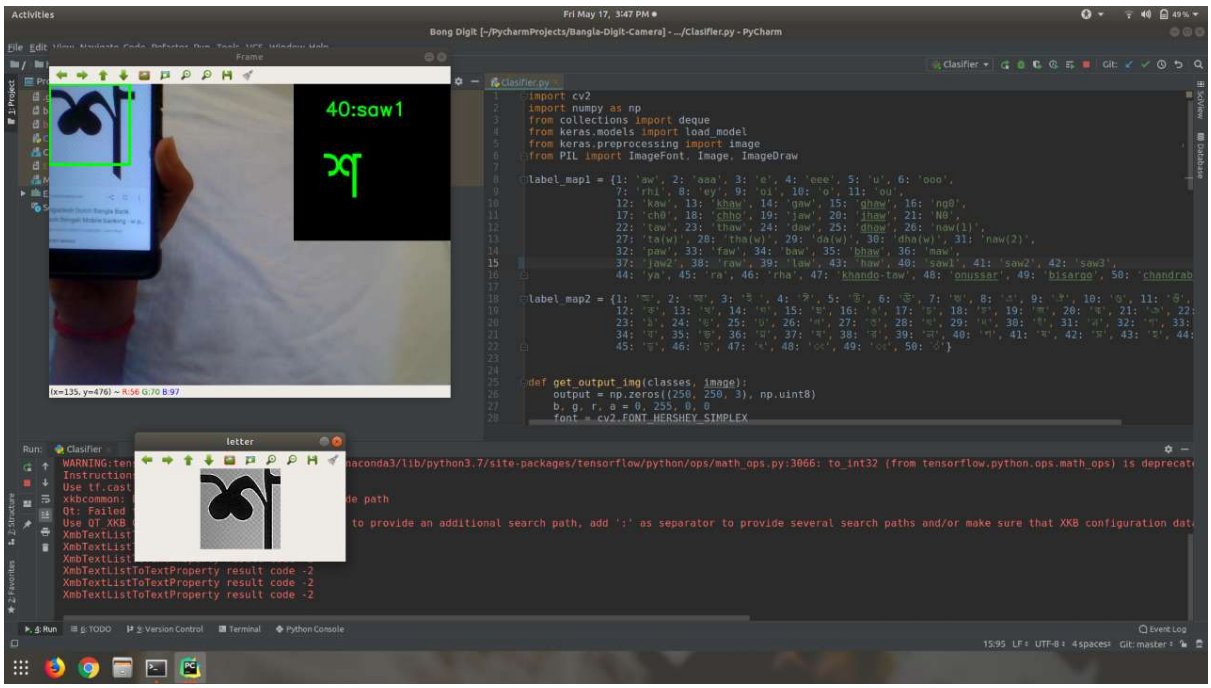


Fig: Demo of second application

Refinement

In traditional machine learning projects, one strives to get a balance between Bias and Variance and its essentially a trade-off. However, with deep learning, bias and variance are orthogonal and can be tuned separately. I have used the following workflow for the refinement:

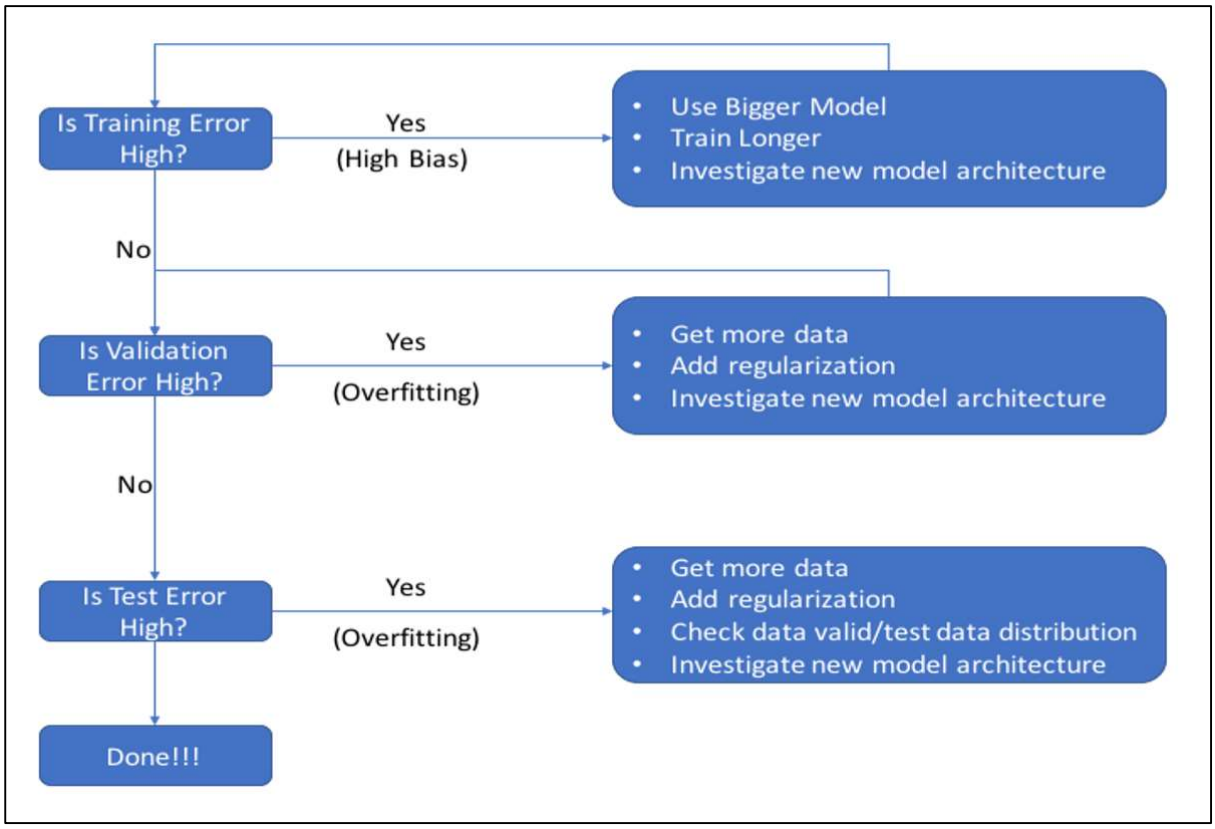


Fig 12: Workflow used for refinement

Tuning the bias:

If the training error is high (higher than the human level) then there are lot of avoidable errors and can be addressed by 1) Bigger Model – Adding more convolution layers to increase the depth of the neural net will ensure that more features are extracted which would improve the accuracy 2) Longer Training – Increasing the number of epochs would ensure that the model explore more weights to find the global minimum 3) New model architecture – Different optimization techniques and varying learning rates, activation functions like leaky ReLU, changing batch size to change the stochasticity, decreasing regularization to allow better fitment, etc

Tuning the variance:

If the validation error is much higher than the training error then that means the model has been overfitted for the training data. To address this we should essentially look at 1) Sourcing more data – Usually more data means more features and the accuracy of prediction improves since the model can generalize better 2) Regularization techniques – Dropout technique that randomly drops neurons in the layers and thereby distributes the weights better is an effective way to address overfitting and thereby ensures that the network is able to generalize better on the validation data 3) Transfer learning whereby the knowledge from another network that addresses a similar problem could be an option If the validation error is low but the testing error is high then there is a possibility that the validation data and testing data are from different distributions. I have ensured that this is ruled out for my project by ensuring same distribution in the two datasets. Apart from the methods listed above Data Augmentation techniques could be used to increase the volume of data. In data augmentation new data is generated from the existing datasets by introducing distortions (rotation, shifting, blurring, zooming etc) in the same image.

Result analysis

Our proposed model gave 86.54% validation accuracy after 13 epochs, 88.37% validation accuracy after 50 epochs and 88.62% validation accuracy after 100 epochs.

1. Epoch 13/100
2. 9600/9600 [==============================] - 43s 5ms/step - loss: 0.1616 - acc: 0.9512 - val_loss: 0.5275 - val_acc: 0.8654

Fig after 13 epochs

1. Epoch 50/100
2. 9600/9600 [==============================] - 43s 5ms/step - loss: 0.0199 - acc: 0.9943 - val_loss: 0.6131 - val_acc: 0.8837

Fig after 50 epochs

1. Epoch 100/100
2. 9600/9600 [==============================] - 43s 5ms/step - loss: 0.0072 - acc: 0.9982 - val_loss: 0.6849 - val_acc: 0.8862

Fig after 100 epochs

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. At the same time, every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent.

Adagrad: Instead of a common learning rate for all parameters, we want to have separate learning rate for each. So Adagrad keeps sum of squares of parameter-wise gradients and modifies individual learning rates using this. As a result, parameters occurring more often have smaller gradients.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

With Adagrad optimizer we got 87.54% accuracy after 13 epochs.

```
1. Epoch 13/13
2. 9600/9600 [==============================] - 1s
   87us/step - loss: 0.0995 - acc: 0.9709 - val_loss: 0.4919 - val_acc: 0.8754
```

Fig: with Adagrad optimizer validation accuracy

RMSProp: In Adagrad, since we keep adding all gradients, gradients become vanishingly small after some time. So, in RMSProp, the idea is to add them in a decaying fashion as

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2$$

Now replace G_t in the denominator of Adagrad equation by this new term. Due to this, the gradients are no more vanishing. With RMSprop optimizer we got 88.17% accuracy after 13 epochs.

```
1. Epoch 13/13
2. 9600/9600 [==============================] - 1s
   89us/step - loss: 0.0741 - acc: 0.9761 - val_loss: 0.5649 - val_acc: 0.8817
```

Fig: with RMSprop optimizer validation accuracy

Adam (Adaptive Moment Estimation): Adam combines RMSProp with Momentum. So, in addition to using the decaying average of past squared gradients for parameter-specific learning rate, it uses a decaying average of past gradients in place of the current gradient (similar to Momentum).

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t$$

The ^ terms are actually bias-corrected averages to ensure that the values are not biased towards 0.

With Adam optimizer we got 88.08% accuracy after 13 epochs.

```
1. Epoch 13/13
2. 9600/9600 [==============================] - 1s
   92us/step - loss: 0.1053 - acc: 0.9654 - val_loss: 0.5159 - val_acc: 0.8808
```

Fig: with Adam optimizer validation accuracy

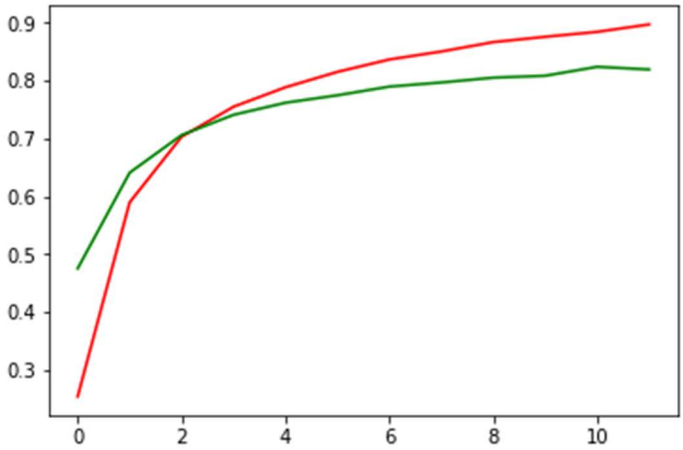


Fig: The graph of training accuracy and validation accuracy after 13 epochs

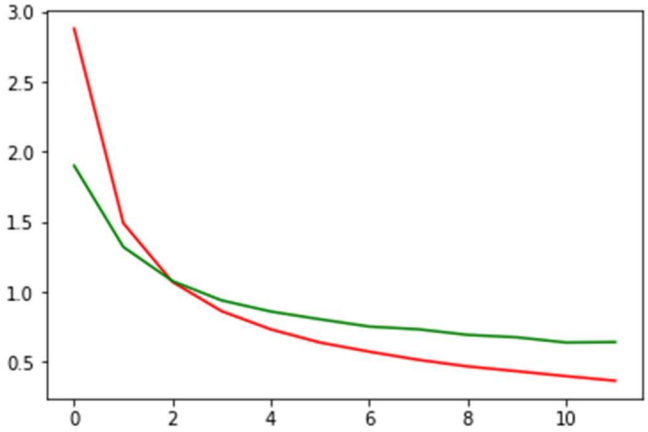


Fig: The graph of training loss and validation loss after 13 epochs

The highest testing accuracy we achieved is 89.26%. We have used Adam optimizer. For different optimizer we would get different values but Adam would be most acceptable.

1. 3000/3000 [==============================] - 0s 77us/step
2. Test loss: 0.6651361061086257
3. Test accuracy: 0.8926666668256124

Fig: Testing accuracy

Conclusion

Convolutional neural network (CNN) has ability to recognize visual patterns directly from pixel images and is the preferred method for image classification problems. Therefore, a CNN structure is investigated without any feature selection for Bangla handwritten pattern classification in this study. The method has been tested on a publicly available handwritten character dataset and outcome compared with existing prominent methods for Bangla. The proposed method is shown to have higher accuracy than any other model to the best of my knowledge.

Further Improvements and Future Research

For improving the model one should add more layer. There are many different models available like AlexNet, VGG net, Lenet, Resnet etc. For improving one should try these convolutional models.

Besides that, our project can only recognize character. One should add LSTM and RCNN to improve so that it can recognize text. For that one has to train model with Bengali number, compound character and matra database so that it can recognize text. As a next step I would like to do these things to explore the word recognition problem in Bengali language.

Bibliography

[1] Chaudhuri, BB and Pal, U. A complete printed bangla ocr system. Pattern recognition, 31(5):531–549, 1998.

[2] O. D. Trier, A. K. Jain and T. Taxt, " Feature extraction methods for character recognition: a survey" Pattern Recognition 29 (4), 641-662. 1996.

[3] S. V. Rajashekararadhya and P. V. Ranjan, "Zone based feature extraction algorithm for handwritten numeral recognition of Kannada script," IEEE International Advance Computing Conference (IACC), Patiala, India, March 2009.

[4] Y. Y. Chung and M. T. Wong, "Handwritten character recognition by Fourier descriptors and neural network," IEEE TENCON, Speech and Image Technologies for Computing and Telecommunications, 1997.

[5] W. Zhang, Y. Y. Tang and Y. Xue, "Handwritten character recognition using combined gradient and wavelet features," International Conference on Computational Intelligence and Security, pp. 662-667, Guangzhou, Nov. 2006.

[6] G. Abandah and N. Anssari, "Novel moment features extraction for recognizing handwritten Arabic letters," Journal of Computer Science, vol. 5, issue 3, pp. 226-232, 2009.

[7] L. Heutte, J. V. Moreau, T. Paquet, Y. Lecourtier, and C. Olivier, "Combining structural and statistical features for the recognition of handwritten characters," Proceedings of 13th International Conference on Pattern Recognition, Vienna, Austria, 1996, Vol. 2, pp. 210-214.

[8] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu and M. Kundu, "Combining multiple feature extraction techniques for handwritten Devnagari character recognition," IEEE Region 10 Colloquium and 3rd International Conference on Industrial and Information Systems, Dec. 2008.

[9] Y. Kimura, A. Suzuki, K. Odaka, "Feature selection for character recognition using genetic algorithm," IEEE Fourth International Conference on Innovative Computing, Information and Control (ICICIC), Kaohsiung , pp. 401-404, Dec. 2009.

[10] A. K. Jain, P. W. Duin, and J. Mao, "Statistical pattern recognition: a review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, Jan. 2000.

[11] A. Pal and D. Singh, "Handwritten English character recognition using neural network," International Journal of Computer Science and Communication, vol. 1, no. 2, pp. 141-144, JulyDec 2010.

[12] J. Pradeep, E. Srinivasan and S. Himavathi, "Diagonal based feature extraction for handwritten alphabets recognition system using neural network," International Journal of Computer Science and Information Technology, vol. 3, no. 1, Feb. 2011.

[13] D. C. Shubhangi and P. S. Hiremath, "Handwritten English character and digit recognition using multiclass SVM classifier and using structural micro features," International Journal of Recent Trends in Engineering, vol. 2, no. 2, Nov. 2009.

[14] D. Nasien, H. Haron and S. S. Yuhaniz, "Support vector machine for English handwritten character recognition," 2nd International Conference on Computer Engineering and Applications, 2010.

[15] D. C. Tran, P. Franco and J. M. Ogier, "Accented handwritten character recognition using SVM: application to French," 12th International Conference on Frontiers in Handwriting Recognition, 2010.

[16] C. L. Liu and M. Nakagawa, "Prototype learning algorithms for nearest neighbor classifier with application to handwritten character recognition," Proc. 5th International Conference on Document Analysis and Recognition (ICDAR), Bangalore, India, Sep. 1999.

[17] B. Feng and X. Ding, "Offline handwritten Chinese character recognition with hidden Markov models," Proc. 5th ICSP, vol. 3, pp. 1542-1545, Beijing, China, 2000.

[18] S. R. Veltman and R. Prasad, "Hidden Markov models applied to online handwritten isolated character recognition," IEEE Transactions on Image Processing, vol. 3, issue 3, pp. 314-318, May 1994.

[19]Heuristic-Based OCR Post-Correction for Smart Phone Applications the university of North Carolina at chapel hill department of computer science honors thesis Author: Wing-Soon Wilson Lian 2009.

[20] Pal, U., Chaudhuri, B.B., Belaïd, A.: A system for bangla handwritten numeral recognition. IETE Journal of Research, Institution of Electronics and Telecommunication Engineers52(1)(2006)

[21] Mandal, S., Sur, S., Dan, A., Bhowmick, P.: Handwritten bangla character recognition in machine-printed forms using gradient information and haar wavelet. In: Image Information Processing (ICIIP), 2011 International Conference on. (2011)

[22] Das, P., Dasgupta, T., Bhattacharya, S.: A novel scheme for bengali handwriting recognition based on morphological operations with adaptive auto-generated structuring elements. In: 2ndInternational Conference on Control, Instrumentation, Energy and Communication (CIEC16).(2016)

[23] A. K. Ray and B. Chatterjee, "Design of a Nearest Neighbor Classifier System for Bengali Character Recognition," *IETE Journal of Research*, vol. 30, no. 6, pp. 226–229, Nov. 1984.

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

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

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

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

[28] U. Bhattacharya, M. Shridhar, and S. K. Parui, "On recognition of handwritten Bangla characters," in *Computer Vision, Graphics and Image Processing*, Springer, 2006, pp. 817–828.

[29] Ahmed Asif Chowdhury, Ejaj Ahmed, Shameem Ahmed, and Shohrab Hossain, "Optical Character Recognition of Bangla Characters using Neural Network: A Better Approach," 2002.

[30] U. Pal and B. B. Chaudhuri, "Automatic Recognition of Unconstrained Off-Line Bangla Handwritten Numerals," 2000, pp. 371–378.

[31] A. R. M. Forkan, S. Saha, M. M. Rahman, and M. A. Sattar, "Recognition of conjunctive Bangla characters by artificial neural network," in *Information and Communication Technology, 2007. ICICT'07. International Conference on*, 2007.

[32] M. A. Hasnat, S. M. Habib, and M. Khan, "A high performance domain specific OCR for Bangla script," in *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics*, Springer, 2008.

[33] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla Numeral Recognition System and Its Application to Postal Automation," *Pattern Recogn.*, vol. 40, no. 1, pp. 99–107, Jan. 2007.

[34] C.-L. Liu and C. Y. Suen, "A New Benchmark on the Recognition of Handwritten Bangla and Farsi Numeral Characters," *Pattern Recogn.*, vol. 42, no. 12, pp. 3287–3295, Dec. 2009.

[35] T. Hassan and H. A. Khan, "Handwritten bangla numeral recognition using local binary pattern," in *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on*, 2015, pp. 1–4

[36] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri, and D. kumar Basu, "An Improved Feature Descriptor for Recognition of Handwritten Bangla Alphabet.

[37] Fukushima, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics, 36(4):193–202, 1980.

[38] LeCun, Yann, Bottou, L´eon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278– 2324, 1998a.

[39] Ciresan, Dan, Meier, Ueli, and Schmidhuber, J¨urgen. Multi-columndeepneuralnetworksforimageclassification. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3642–3649. IEEE, 2012.

[40]Ciresan, D. and Meier, U. Multi-column deep neural networks for offline handwritten chinese character classification. In International Joint Conference on Neural Networks (IJCNN), pp. 1–6, July 2015.

[41] Gradient Based Learning Applied to Document Recognitiion, Yan LeCun, Patrick Haphner, Leon Bottuoand Yoshua Bengio IEEE Nov 1998

[42] Halima Begum et al, Recognition of Handwritten Bangla Characters using Gabor Filter and Artificial Neural Network, International Journal of Computer Technology & Applications, Vol 8(5),618-621, ISSN:2229-6093

[43] Nibaran Das, Sandip Pramanik, "Recognition of Handwritten Bangla Basic Character and Digit Using Convex Hall Basic Feature". 2009 International Conference on Artificial Intelligence and Pattern Recognition(AIPR-09)

[44] Md. M. Rahman, M. A. H. Akhand, S. Islam, P. C. Shill, "Bangla Handwritten Character Recognition using Convolutional Neural Network," I. J. Image, Graphics and Signal Processing, vol. 8, pp. 42-49, July 15, 2015.

[45] N. Das 1, B. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, "Handwritten Bangla Basic and Compound character recognition using MLP and SVM classifier," Journal of Computing, vol. 2, Feb. 2010.

[46] 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)*, 2017, pp. 1–6.

[47] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," in *Computer and Information Technology (ICCIT), 2017 20th International Conference of*, 2017, pp. 1–5.

[48] M. Z. Alom, P. Sidike, M. Hasan, T. M. Taha, and V. K. Asari, "Handwritten Bangla Character Recognition Using the State-of-the-Art Deep

Convolutional Neural Networks," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, Aug. 2018.

[49] Zohra Saidane and Christophe Garcia. Automatic scene text recognition using a con-volutional neural network. InWorkshop on Camera-Based Document Analysis andRecognition, 2007

[50] U. Meier, D.C. Ciresan, L.M. Gambardella, and J. Schmidhuber. Better digit recogni-tion with a committee of simple neural nets. InICDAR, 2011

[51] Christopher M. Bishop.Pattern Recognition and Machine Learning. Springer, NewYork, NY, USA, 2006

[52] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recog-nition. InComputer Vision and Pattern Recognition, 2010.

[53] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, andL. D. Jackel. Backpropagation applied to handwritten zip code recognition.NeuralComputation, 1:541–551, 1989

[54] S. Roy, N. Das, M. Kundu, and M. Nasipuri, "Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach," *Pattern Recognition Letters*, vol. 90, pp. 15–21, Apr. 2017.

[55] M. M. Rahman, M. A. H. Akhand, S. Islam, P. Chandra Shill, and M. M. Hafizur Rahman, "Bangla Handwritten Character Recognition using Convolutional Neural Network," *International Journal of Image, Graphics and Signal Processing*, vol. 7, no. 8, pp. 42–49, Jul. 2015.

[56] S. M. A. Sharif, N. Mohammed, N. Mansoor, and S. Momen, "A hybrid deep model with HOG features for Bangla handwritten numeral classification," *2016 9th International Conference on Electrical and Computer Engineering (ICECE)*, pp. 463–466, 2016.

[57] Nibaran Das, Subhadip Basu, PK Saha, Ram Sarkar , Mahantapas Kundu , and Mita Nasipuri , "Handwritten Bangla Character Recognition Using a Soft Computing Paradigm Embedded in Two Pass Approach," *Pattern Recognition*, vol. 48, no. 6, pp. 2054–2071, Jun. 2015.