*I.J. Modern Education and Computer Science*, **2020, 2, 19-29**
Published Online April 2020 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2020.02.03

# OCR for Printed Bangla Characters Using Neural Network

**Asif Isthiaq**
Department of Computer Science and Engineering, Ahsanullah University of Science and Technology,
141&142, Love Road, Tejgaon Industrial Area, Dhaka 1208, Bangladesh
Email: asifisthiaq@gmail.com

**Najoa Asreen Saif**
Department of Computer Science and Engineering, Ahsanullah University of Science and Technology,
141&142, Love Road, Tejgaon Industrial Area, Dhaka 1208, Bangladesh
Email: najoa06@gmail.com

Received: 28 October 2019; Accepted: 23 November 2019; Published: 08 April 2020

***Abstract*—**Optical Character recognition is a buzzword in the field of computing. Artificial neural networks are being used to recognize characters for a long time. ANN has the ability to learn and model non-linear and complex relationships, which is really important because in real life, many of the relationships between inputs and outputs are non-linear as well as complex. Research in the field of OCR with Bangla language is not as vast as the English language. So, there is a scope of research in this area. It can be used to search and scan hundreds of Bangla documents within seconds and can easily manipulate the data. It is developed for various purpose like for vision impaired person where OCR software can help turn books, magazines and other printed documents into accessible files that they can listen. The limitation of traditional OCR are sufficient dataset is not available, all different font of characters are not available and there are lots of complex and similar shape characters for which accuracy not good. In our research, we first tried to make a dataset large enough so that we can train our neural network as they require big data to train. We built our own dataset of 2,97,898 Bangla single character images of different fonts . Then for implementing neural network we used Scikit-learn's multi-layer perceptron classifier and we also implemented our own multi-layer feed forward back propagation neural network using a machine learning framework named Tensorflow. We have also built a GUI application to demonstrate the recognition of Bangla single character images.

***Index Terms*—**Dataset Creation, Neural Network, Training Model, Testing Model, Classification .

## I. INTRODUCTION

Character recognition is easy for humans. We learn to recognize the differences between letters of a language from a young age. We train our mind to recognize between letters, words and figures from childhood. In optical character recognition, scientists have been trying to implement methods which are used by human brains to recognize characters. neural network is one of them. It was made trying to duplicate the human brain. We built our own dataset and implemented a neural network for training and testing. First we studied to find out what type of work has been done in the field of character recognition and we have found some limitations within the existing research regarding Bangla OCR. So, in order to provide a better approach we have created our own dataset of Bangla single characters. Then we have implemented our multilayer feedforward back propagation neural network with different approaches. We have also built a GUI application for the demonstration of recognition of Bangla single characters in which the number of hidden layer and number of neurons in those hidden layers can be set by the user.

Optical Character Recognition or OCR is a technology which is used to recognize characters from a digital image or scanned documents by examining the text of a digital image or document and translate the characters in such a way that can be manipulated and used for data processing. In our research we have worked with printed characters.

There are several phases to recognize the characters successfully. First, the image needs to undergo some pre-processing steps such as noise reduction, binarization, skew detection and correction etc. Next is segmentation phase where lines are segmented into words and words into characters. Then set of features are extracted from those segmented character images in the feature extraction phase. At the end segmented character image is classified using a classifier in the classification phase.

Bangla OCR can be used in many sectors. One may able to search hundreds of Bangla documents, books etc. and locate what they need within seconds using the digital search features. Bangla OCR allows scanning of Bangla documents and make it available to manipulate easily and relevant information can be added later on. Scanned documents can be stored in digital form so lot of space can be saved and so backup can be created to

Copyright © 2020 MECS 

prevent it from any loss. Bangla OCR applications can be used for various purposes like if someone is vision impaired, OCR software can help turn books, magazines and other printed documents into accessible files that they can listen.

There are several limitations existing in Bangla OCR. Sufficient dataset not available for Bangla printed characters. Sufficient dataset with different fonts is not available for Bangla printed characters. Due to complexity and similarities between different characters accuracy is not good enough.

Our objective is to create Bangla single character dataset, developing a multilayer Feedforward Backpropagation Neural Network classifier for Bangla OCR and creating a model by which we can recognize any given character image of any size just by resizing the character into 40 by 40 pixels.

As our data is complex, we are motivated to use neural network because it gives better result.

To create dataset we have taken 10 different bangla fonts of 40*40. Then use the Matlab code to generate all possible position from the cropped image. We also used circshift function to shift pixels from place to place so that we can get good amount of picture from single dataset.

Neural networks are a set of algorithms that are designed to recognize patterns. We have implemented neural network using Sklearn MLP Classifier and using Tenserflow.

In Sklearn MLP Classifier, we used model with one hidden layer and two hidden layers. In those hidden layers we have used different numbers of neuron to check the accuracy of our model. Two approaches was done when implementing this classifier. When selecting the train and test data we have used two different ways.

First we used our own dataset and used the train test split method to divide the dataset into train and test data.

For the second approach we used a new font named turagmj as a test data and the whole dataset is used as training data.

We implemented neural network using TensorFlow with both one and two hidden layers. In this implementation of neural network we have used three layers. One input layer, one hidden layer and one output layer.

## II. Related Works

Some work has been done regarding OCR and we have been inspired by this. In this Table 1 we have discussed some papers we have read and their approaches.

Table 1. Some related work regarding OCR

| Work | Summary |
|---|---|
| Optical Character Recognition(OCR) for printed Devnagari Script Using Artificial Neural Network [1] | The researchers proposed a technique for OCR system for printed Devnagari scripting artificial neural network with back propagation algorithm with two hidden layers. Input matrix of size (48×57) had given them the best result. For classification of characters they had considered three features which are mean distance, histogram of projection based on spatial position of pixel and histogram of projection based on pixel value. |
| Optical Character Recognition Using Artificial Neural Network [2] | In research [2], they described an offline handwritten alphabetical character recognition system using multilayer feed forward neural network. The whole process of recognition included two phases training and testing. Both the phases consisted of pre-processing, feature extraction and classification. The network had 35 neurons. |
| Application of Neural Networks in Character Recognition [3] | In research[3], they used neural networks in recognizing characters from a printed script. The algorithm used was BPN (Back Propagation Neural Network). The BP algorithm determines the weight for a multilayer ANN with feed-forward connections .The activation function used was Logsig. |
| Optical Character Recognition using Back Propagation Neural Network [4] | In research [4], they used back propagation neural network to recognize printed English characters. The feedforward network had one input, one hidden layer and one output layer. The network was trained with 558 samples of 62 characters where each character had 9 samples. They tested their trained network with more than 10 samples per character and gave 99% accuracy for numeric digits (0-9), 97% accuracy for capital letters (A-Z), 96% accuracy for small letters (a-z). |
| Different methods for Optical Character Recognition [5] | In research [5], they described the different methods for optical character recognition. The authors covered different phases to develop a complete OCR but they emphasize on the feature extraction process. The accuracy of their combined method provided 90% accuracy in some cases. They had used multilayer neural network as their classifier. |
| Implementing Recognition of hand written characters [6] | In research [6], they gave a good idea of implementing recognition of hand written characters. In this research work the authors made the dataset written by 1000 people. They used Support Vector Machine which is used for classification in pattern recognition. This RBF SVM produced 93.43% overall recognition rate. |
| Different models of Neural Network to find the effectiveness of the Neural Network [7] | In research [7], they used different models of neural network and they recorded certain parameters to find the effectiveness of the neural network. They used image size (10×10) as an input to the neural network. |
| Minimally Segmenting High Performance Bangla Optical Character Recognition Using Kohonen Network [8] | In research [8], they described a method to recognize Bangla character using kohonen neural network. The images are first converted into gray scale and then to binary images. Then these images are scaled to fit a pre-determined area with a fixed but significant number of pixels. The feature vectors are then extracted. Finally a kohenen neural network is chosen for the training and classification process. The resulting classifier was accurate in recognizing characters better than 98% depending on the quality of the input images. |
| A Complete Bangla OCR System for Printed Characters [9] | In research [9], it gave us the comprehensive description of different phases of OCR and how to recognize characters using artificial neural network. Preprocessing steps of this article includes binarization, noise removal, skew detection and correction, segmentation in various levels and scaling. The output of neuron for each character was 50. |
| Recognition of Conjunctive Bangla Characters by Artificial Neural Network [10] | In research [10], they tried to recognize Bangla conjunctive characters. The method used a comparison between sample data and training data components. Each character patterns scaled onto(16×16) binary image. So the input layer had 256 inputs. The network had one hidden layer. The hidden layer neurons was estimated to be 100. The output layer was 193. They had worked with 100 samples. |

## III. Implementation of Our Proposed Work

### A. Our Proposed Work

(1) Creating Bangla single characters dataset so that after us others can use it which would give them better recognition rate. We took images of (40 * 40) 1600 pixels.

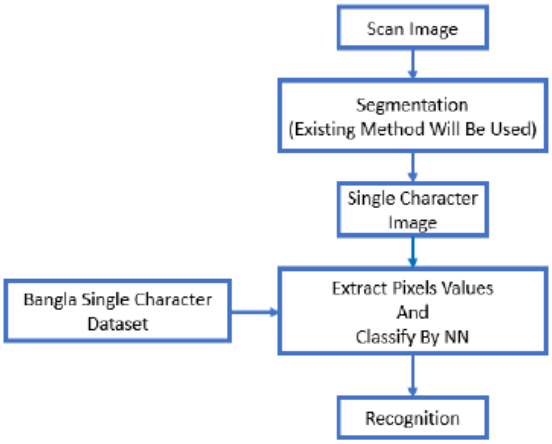


Fig. 1. Phases of our proposed OCR

(2) At first, we wanted to take (20 * 20) 400 pixels image of each single character but accuracy was poor. So we changed the image size to (40 * 40) 1600 pixels for better accuracy.
(3) Implementing a feed forward back propagation neural network using different approaches.
(4) Using different activation function in hidden layers of neural network and compare which one will give better accuracy.

### B. Dataset Creation

We need to build a Bangla single character's dataset from scratch. We have built a dataset of 2,97,898 images. We took 50 Bangla characters form 'a' to 'ri' and from 'ko' to 'khandata'. For each character we took 10 Bangla fonts. They are:

(1) BhrahmaputraMJ
(2) BorhalMJ
(3) ChandrabatiMatraMJ
(4) DhorolaMJ
(5) GangaMJ
(6) GoomtiMJ
(7) KongshoMatraMJ
(8) ModhumatiMJ
(9) PadmaMJ
(10) RatoolMJ

There is an image of "ka" in 10 different fonts in Fig. 2.


Fig. 2. "ka" in 10 different fonts

We have also written a Matlab code which would enable us to make many images from one image. We have mainly done that to make many instances of a character and to take letter of a font to different places so that the neural network gets a higher chance of better training and the accuracy can increase. Let us think that we have just drawn a character named 'ko' with the help of Photoshop. The steps after that is given below.

1) **Step 1:** Cut the image by any snipping app like Snipping Tool or PicPick to create a 40*40 image. Each character in Bangla letters has different height and width so we selected a fixed font size so that the character with maximum height and width can be cropped using 40 * 40 window size.
2) **Step 2:** Then use the Matlab code to generate all possible position from the cropped image. We do this because we wanted to increase the number of images in the dataset. In the code we have function named top, bottom, right and left. First top and bottom function is called to generate all possible top and bottom positions. Then right function is called so the image is right shifted by one pixel and then top and bottom function is called so that all possible top and bottom position image is generates from that shifted image until the shifted image hit the boundary pixels. We do the same process when left function is called.

In Fig. 3 we can see that the first image of 'ko'. It is the picture we get after cropping using Picpick from photoshop. Then after feeding the image to our code we get all possible shifted position of that image in Fig. 3 are some images that are generated after feeding the image to the Matlab code.

Copyright © 2020 MECS 

Fig. 3. Single image into multiple images

We have used circshift function to shift the pixels from place to place. So we get good amount of pictures from a single image and that's why the data set can be much larger and the neural network gets a big enough dataset to train. Fig. 4 is the result when applying the rotate function.

Fig. 4. Rotation of a character image

In Fig. 4 the first and second row gives us the rotated image. First row gives us 15, 45, 60 and 90 degree rotated version of the image. On the Second row we get the -15, -45, - 60 and -90 degree rotated version of that image.

3) **Step 3:** After applying the same process for all the letters and all the fonts we got our image dataset. Then we labeled out dataset where "a" is denoted by 1 and so on and lastly "khandacta" is denoted by 50 and save it in a csv file.
4) **Step 4:** This will be the final step of our dataset creation process. For that we wrote another code named DatasetCreation.py. It is written using python programming language. The images up to this point is two dimensional and 40*40. Now we convert this image into one dimension which is 1*1600. First we did the usual preprocessing like rgb to gray conversion. Then we convert the grey image to binary image which means the image will only contain black and white pixels. Finally, we convert the two-dimensional image into a one-dimensional image and then the image is appended in the dataset.csv. So one row of the dataset gives us the one character or letter. So, in the csv file there is 1600 columns and 2,97,898 rows. Each row represents one letter with its own class labeled. After doing all the processes mentioned above, we get our final dataset. And now our dataset is ready to be used by the neural network.

### *C. Description Of The Dataset*

Table I gives us a clear idea about our dataset. We took 10 fonts of a single character and draw those fonts in different styles like bold, smooth etc. Then we created a vast dataset by shifting those characters in different positions. Our dataset holds total of 2,97,898 images. We took 40*40 images.

Table 2. Number of character image in each class

| Serial No. | Character Name | Total Number of Character Image in Each Class |
|---|---|---|
| 1 | অ | 1973 |
| 2 | আ | 386 |
| 3 | ই | 2264 |
| 4 | ঈ | 2012 |
| 5 | উ | 2588 |
| 6 | ঊ | 2330 |
| 7 | ঋ | 7712 |
| 8 | এ | 2204 |
| 9 | ঐ | 8453 |
| 10 | ও | 2639 |
| 11 | ঔ | 4262 |
| 12 | ক | 5427 |
| 13 | খ | 8323 |
| 14 | গ | 8453 |
| 15 | ঘ | 17259 |
| 16 | ঙ | 18663 |
| 17 | চ | 2882 |
| 18 | ছ | 1849 |
| 19 | জ | 1372 |
| 20 | ঝ | 3562 |
| 21 | ঞ | 3327 |
| 22 | ট | 8593 |
| 23 | ঠ | 6746 |
| 24 | ড | 8289 |
| 25 | ঢ | 5104 |
| 26 | ণ | 3260 |
| 27 | ত | 3311 |
| 28 | থ | 3294 |
| 29 | দ | 4914 |
| 30 | ধ | 2549 |
| 31 | ন | 2484 |
| 32 | প | 11641 |
| 33 | ফ | 6709 |
| 34 | ব | 11786 |
| 35 | ভ | 8715 |
| 36 | ম | 11462 |
| 37 | য | 10454 |
| 38 | র | 7271 |
| 39 | ল | 4832 |
| 40 | শ | 3969 |
| 41 | ষ | 4629 |
| 42 | স | 1644 |

| | | |
|---|---|---|
| 43 | হ | 3024 |
| 44 | ড় | 3133 |
| 45 | ঢ় | 4441 |
| 46 | য় | 2623 |
| 47 | □ | 16403 |
| 48 | □ | 8759 |
| 49 | □ | 15722 |
| 50 | ৎ | 4148 |

*D. Implementation Using Sklearn MLP Classifier*

We have first implemented our neural network using MLP classifier. MLP's full form is multi-layer perceptron classifier. This model optimizes the log-loss function using stochastic gradient descent. We have used model with both one hidden layer and two hidden layers. In those hidden layers we have used different numbers of neuron to check the accuracy of our model. We have done our coding in python language. We have used two approaches when implementing this classifier. When selecting the train and test data we have used two different ways. First we used our own dataset and used the train test split method to divide the dataset into train and test data. We had given 20% for test and 80% for the training process. For the second approach, we used a new font named turagmj as a test data and the whole dataset is used as training data. Both of the approaches are similar. For train test split we need to read the data and label of the data then split the data in train and test. But for the second approach both our own data and the new test data which is totally unfamiliar needs to be read separately with their label. We have train and test our data with using both 1 hidden layer and 2 hidden layers. So the steps are as follows:

(1) First, we imported the necessary libraries.
(2) Then we read the dataset.
(3) Next step was to analyze the dataset to make it useful for our work.
(4) The step after that was to prepare the data. Trainand test data each are divided into two subdivision for train test split. For using a totally different test data split is not required.
(5) Then the train values are scaled so that the values which are large cannot affect the outcome that much.
(6) Finally, we train and test the dataset using the MLPClassifier.

*E. Implementation Using Tensorflow*

Machine learning frameworks like TensorFlow, PaddlePaddle, Torch, Caffe, Keras, and many others can speed up machine learning development significantly. Programing frameworks can not only shorten coding time but sometimes also perform optimizations that speed up code. In this implementation, we have used TensorFlow. First we initialize the variables then we start our own session then we train algorithm and then implement a neural network,

Writing and running programs in TensorFlow has the following steps:

(1) Create Tensors (variables) that are not yet executed/evaluated.
(2) Write operations between those Tensors.
(3) Initialize the Tensors.
(4) Create a Session.
(5) Run the Session.

When we created a variable, we simply defined the variable , but did not evaluate its value. To evaluate it, we had to run:

init = t .global_variables_initializer()

That initialized the variable, and then we finally able to evaluate the value of the variable. Let us look at an easy example :

```
a = tf.constant(2)
b = tf.constant(10)
c = tf.multiply(a,b)
print(c)
```

```
Tensor("Mul:0", shape=(), dtype=int32)
```

Fig. 5. An example of variable initializing

As expected, the result is not 20. We got a tensor saying that the result is a tensor that does not have the shape attribute, and is of type "int32". All we did was put in the *computation graph, but we have not run this computation yet. In order to actually multiply the two numbers, we will have to create a session and run it.

```
sess = tf.Session()
print(sess.run(c))
```

```
20
```

Fig. 6. An example of session

Next we use placeholders . A placeholder is an object whose value we can specify only later. To specify values for a placeholder, we can pass in values by using a "feed dictionary" (feed_dict variable). Below, we created a placeholder for x. This allows us to pass in a number later when we run the session.

```
x = tf.placeholder(tf.int64, name = 'x')
print(sess.run(2 * x, feed_dict = {x: 3}))
sess.close()
```

```
6
```

Fig. 7. An example of Placeholders

When we first defined x we did not have to specify a value for it. A placeholder is simply a variable that we will assign data to only later, when running the session. We can say that we feed data to these placeholders when running the session.

Here's what's happening: When we specify the operations needed for a computation, we are telling TensorFlow how to construct a computation graph.

The computation graph can have some placeholders whose values we will specify only later. Finally, when we

run the session, we are telling TensorFlow to execute the computation graph.

Next, to implement a linear function we will need the following functions :

– tf.matmul(..., ...) to do a matrix multiplication
– tf.add(..., ...) to do an addition

Next, we implemented one-hot encoding on categorical data. Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Categorical variables are often called nominal. Some examples include:

– A “pet” variable with the values: “dog” and “cat”.
– A “color” variable with the values: “red”, “green” and “blue”.
– A “place” variable with the values: “first”, “second” and “third”.

Each value represents a different category. Some categories may have a natural relationship to each other, such as a natural ordering. The “place” variable above does have a natural ordering of values. This type of categorical variable is called an ordinal variable.

Some algorithms can work with categorical data directly. For example, a decision tree can be learned directly from categorical data with no data transform required (this depends on the specific implementation). Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves. This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application. For categorical variables where no such ordinal relationship exists, the integer encoding is not enough. In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories). In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer .

```
import numpy as np
from sklearn.preprocessing import OneHotEncoder

target = np.array([1,2,3])
target = target.reshape([-1, 1])
encoder = OneHotEncoder(sparse=False)
encoder.fit(target)
encoded = encoder.transform(target)
print(encoded)
```

```
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
```

Fig. 8. An example of one hot encodinig

We implemented neural network using TensorFlow with both one and two hidden layers.

In this implementation of neural network, we have used three layers. One input layer, one hidden layer and one output layer. Input layer has 1600 nodes or neurons. In the hidden layer we have used different number of nodes or neurons to check the accuracy of our model and in our output layer we have 50 nodes or neurons.

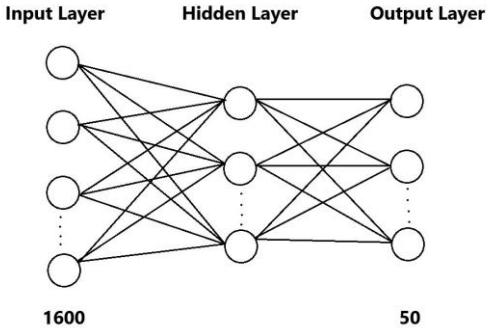


Fig. 9. Our Neural Network with One Hidden Layer

For this implementation we have done our coding in python. The steps are as follows:

(1) First we need to import all necessary libraries.
(2) Next we load the dataset from a csv file with labels. Then we split the dataset between train and test. For doing so a function is called named load dataset() which adds the bias to the data. Then it converts the numerical target data or label to binary data using one hot encoding. Then the function returns the splitted dataset. We had given 20% for test and 80% for the training process.
(3) Then we specify the numbers of neurons in each layer. Here the number of neurons in the input layer and output layer is fixed which is 1600 and 50 consecutively. We vary the number of neurons in the hidden layer to check the accuracy of our model.
(4) Then we create placeholders for our data.
(5) Then weight initialization is done by calling init_weights() function. This function uses tfrandom_ normal() to give a random weight to each edge or synapses.
(6) After initialization of weights the forwardprop() function is called. This function multiply the inputs with their corresponding weights using tf.matmul() function. Then sigmoid activation function is applied to the result of the matrix multiplication.
(7) Next the backpropagation part starts. The cost is calculated using the softmax cross entropy with logits( ) function. Update is done by using GradientDescentOptimizer() function.
(8) Lastly, we feed the all the training examples and with their corresponding labels using feed_dict and calculate training and testing accuracy using actual and predicted value.

Copyright © 2020 MECS 

In this implementation of neural network we have used four layers. One input layer, two hidden layers and one output layer. Input layer has 1600 nodes or neurons. In the two hidden layers we have used different number of nodes or neurons to check the accuracy of our model and in our output layer we have 50 nodes or neurons. For this implementation we have done our coding in python. Our goal is to build an algorithm capable of recognizing a Bangla character with high accuracy. To do so, we are going to build a tensorflow model using a softmax output. The model is:

*LINEAR -> RELU-> LINEAR -> RELU -> LINEAR-> SOFTMAX*

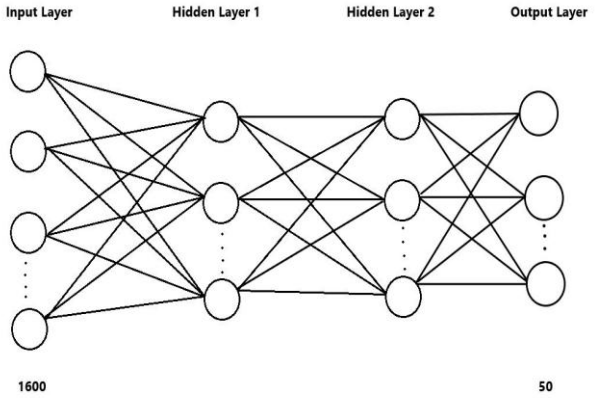


Fig. 10. Our Neural Network with Two Hidden Layer

In the output layer we will use softmax activation function. A softmax layer generalizes sigmoid to when there are more than two classes. The steps of building our model are as follows:

(1) Our first task is to create placeholders for training examples and their corresponding labels. In our dataset the size of each the training example is 1600 and there are 50 classes. This will allow us to later pass our training data in when we run our session. For this task we have implemented a function called create placeholders() to create the placeholders in tensorflow.

(2) Our second task is to initialize the parameters in tensorflow. For this task we have implemented a function called initialize parameters() to initialize the parameters in tensorflow. We are using Xavier Initialization for weights and Zero Initialization for biases.

(3) Then we have implemented the forward propagation module in tensorflow. The function will take in a dictionary of parameters and it will complete the forward pass. This function multiply the inputs with their corresponding weights using tf.matmul() function. Then relu activation function is applied to the result of the matrix multiplication.

(4) Next we calculate the cost using the softmax cross entropy with logits( ) function by implementing a function compute cost().

(5) After we compute the cost function we have created an "optimizer" object. We have to call this object along with the cost when running the tf.session. When called, it will perform an optimization on the given cost with the chosen method and learning rate. In our implementation we chose gradient descent as our method and 0.0001 as our learning rate. Then the update is done by using AdamOptimizer() function. Backpropagation is computed by passing through the tensorflow graph in the reverse order. From cost to inputs.

(6) Next we have implemented our model. We train our model with 10 different fonts of Bangla language and tested our model with a test dataset of completely different font that is not in the training dataset.

## IV. Result and Discussion

To discuss our result we need to talk about our approach of building the dataset because different datasets gave us different result. First we thought of building our dataset 20 * 20. For this first we cut images of letters with size 200 * 200 and then resize it in 20 * 20. For testing first we made the dataset only for 3 letters. We took 3 sizes and 10 fonts. We made 13,245 images from those 3 letters. When we tested the accuracy we found the accuracy to be little bit more than 80%. When we made the same dataset for 10 letters the accuracy dropped to 60%. So we knew our approach needed to be changed. Then we thought of not resizing the image because it degrades the image quality. We changed the size from 20 * 20 to 40 * 40. So we cut the images in 40 * 40. We took 10 fonts.

### *A. Training & Testing Using Train Test Split*

We have a dataset of 2,97,898 images. We divided the dataset in to five parts and we train our neural network with four of those five parts and test neural network with the remaining one part. In other words, we took 80% as train data and 20% as test data of our dataset.

We have 1600 input neurons in the input layer and 50 output neurons in the output layer.

#### *Accuracy Using Sklearn MLP Classifier With 1 HL*

After testing with MLP classifier with one hidden layer with different number of neurons in the hidden layer we get the training and testing accuracy shown in Table 3.

Table 3. Training and Testing Accuracy using MLP with One Hidden Layer

| Hidden Layer 1 | Training Accuracy | Testing Accuracy |
|---|---|---|
| 40 Neurons | 97.15% | 91.61% |
| 50 Neurons | 99.01% | 95.34% |
| 60 Neurons | 99.07% | 95.78% |
| 100 Neurons | 99.21% | 97.07% |

***Accuracy Using Our Implemented Neural Network With 1 HL.***

After testing with neural network, we have implemented with one hidden layer using google's Tensorflow framework with different number of neurons in the hidden layer we get the training and testing accuracy shown in Table 4.

Table 4. Training And Testing Accuracy using our NN With One Hidden Layer

| Hidden Layer 1 | Training Accuracy | Testing Accuracy |
|---|---|---|
| 40 Neurons | 97.89% | 90.19% |
| 50 Neurons | 98.15% | 94.44% |
| 60 Neurons | 99.03% | 95.06% |
| 100 Neurons | 99.12% | 96.34% |

***Accuracy Using Sklearn MLP Classifier With 2 HL.***

After testing with MLP classifier with two hidden layer with different number of neurons in the hidden layers we get the following training and testing accuracy.

Table 5. Training And Testing Accuracy using MLP With Two Hidden Layer

| Hidden Layer 1 | Hidden Layyer 2 | traing Accuracy | Testing accuracy |
|---|---|---|---|
| 40 Neurons | 20 Neurons | 97.07% | 91.77% |
| 50 Neurons | 25 Neurons | 98.77% | 94.02% |
| 60 Neurons | 30 Neurons | 98.06% | 94.73% |
| 100 Neurons | 50 Neurons | 99.14% | 98.13% |

From the above accuracy Table 5 we can see that accuracy increases when we use two hidden layers instead of one. We can also see that accuracy increases when we increase the number of neurons in the hidden layers. When we used 100 neurons in the first hidden layer and 50 neurons in the second hidden layer, we get 98.13% testing accuracy. For this particular neural network testing accuracy of each individual class is given in Table 6.

Table 6. Accuracy of each class

| Serial No. | Character Name | Accuracy of Each Class |
|---|---|---|
| 1 | অ | 98.84% |
| 2 | আ | 100.0% |
| 3 | ই | 99.36% |
| 4 | ঈ | 98.70% |
| 5 | উ | 98.05% |
| 6 | ঊ | 99.55% |
| 7 | ঋ | 97.54% |
| 8 | এ | 100.0% |
| 9 | ঐ | 99.19% |
| 10 | ও | 99.63% |
| 11 | ঔ | 99.88% |
| 12 | ক | 93.54% |
| 13 | খ | 97.67% |
| 14 | গ | 98.74% |
| 15 | ঘ | 97.99% |
| 16 | ঙ | 99.36% |
| 17 | চ | 99.03% |
| 18 | ছ | 97.48% |
| 19 | জ | 97.83% |
| 20 | ঝ | 98.20% |
| 21 | ঞ | 99.41% |
| 22 | ট | 98.76% |
| 23 | ঠ | 99.31% |
| 24 | ড | 94.04% |
| 25 | ঢ | 99.02% |
| 26 | ণ | 98.25% |
| 27 | ত | 96.44% |
| 28 | থ | 97.74% |
| 29 | দ | 100.0% |
| 30 | ধ | 100.0% |
| 31 | ন | 99.39% |
| 32 | প | 99.59% |
| 33 | ফ | 98.23% |
| 34 | ব | 98.75% |
| 35 | ভ | 97.52% |
| 36 | ম | 97.64% |
| 37 | য | 92.48% |
| 38 | র | 95.76% |
| 39 | ল | 99.39% |
| 40 | শ | 99.25% |
| 41 | ষ | 94.07% |
| 42 | স | 96.71% |
| 43 | হ | 99.49% |
| 44 | ড় | 95.29% |
| 45 | ঢ় | 97.42% |
| 46 | য় | 98.82% |
| 47 | □ | 99.91% |
| 48 | □ | 97.89% |
| 49 | □ | 99.49% |
| 50 | ৎ | 99.01% |

*B. Testing Using Different Font*

We also tried to test the accuracy using a totally different font that is not present in our dataset. For that we took a Bangla font turagMJ. We have 6171 as our test data. So, our train data size is 2,97,898 and our test data size is 6171. We used both MLP classifier with two hidden layers and our implemented neural network using google's tensorflow framework with two hidden layers to check the accuracy after testing with our test data. The number of input neuron is 1600 and the number of output neuron is 50. We used two hidden layers in our neural network. We used different combination of neurons in both the hidden layers. 400 neurons in hidden layer 1 and 200 neurons in hidden layer give the best result. As we can see in Table 7 the result is not that great. Analyzing the result, we can say that because of the complexness, similarity and variation between fonts of Bangla characters it is very hard for the neural network to correctly classify them. Some of the characters accuracy is high but some of the characters give very low accuracy. So some additional preprocessing step is needed to increase the accuracy further. We think if we apply some morphological operation on the character images before feature extraction it will increase the chances of recognition.

Copyright © 2020 MECS 

Table 7. Training and Testing accuracy using MLP with 2 hidden layer

| Model Name | Neurons In HL | Training Accuracy | Testing Accuracy |
|---|---|---|---|
| MLP | 400,200 | 99.38% | 71.23% |
| Implemented NN | 400,200 | 99.44% | 69.82% |

## V. Our GUI Application

OCR (optical character recognition) is the use of technology to distinguish printed or handwritten text characters inside digital images of physical documents, such as a scanned paper document. The basic process of OCR involves examining the text of a document and translating the characters into code that can be used for data processing. OCR is sometimes also referred to as text recognition. First the image needs to undergo some pre-processing. Next is segmentation phase where lines are segmented into words and words into characters. Then set of features are extracted from those segmented character images in the feature extraction phase. At the end segmented character image is classified. We wanted to develop an application to give a demonstration of optical character recognition. We have built a desktop application with python scripting language. This application uses Multilayer Perceptron classifier to classify. This application is capable of classifying characters using both 1 hidden layer and 2 hidden layers. This application allows to change the number of neurons in the hidden layers.

Our application gives a demonstration of optical character recognition. To develop our application, we first need to import some necessary libraries such as Tkinter and PIL. We have also created two modules named FeatureExtraction and MLP. We also need to import these two modules. Below there is a snapshot of our application when we first run it.

To test our application, we first need to choose a Bangla character image of resolution 40*40. We can do it in two ways. We can do it by pressing the browse button or clicking the menu File. After clicking the file menu, we can see two option one is open which is used for choosing a character image of resolution 40*40 and the other option is exit by clicking exit we can exit from the application. After choosing a character image the image will appear in the choose character image section.

After choosing a character image for testing we have to extract features from that selected image. We have done it using the FeatureExtraction module. First, we convert the color image into a grey scale image. Then we convert the grey image into binary image. White pixels are denoted as 1 and black pixels are denoted as 0. Then we extract features from that binary image and save it in a csv file.

After extracting feature from selected character image now we want to train our model. For this we used MLP module. We can specify the number of neurons in the hidden layer by giving it as an input in the entry section of our application. If we just enter the number of neurons it will be considered that our neural network will have only one hidden layer. Then after pressing the Train & Test button our model will be trained and the model will be tested using the csv file which we created by extracting features from the selected image. After training & testing the selected character image is predicted.

If we want to train & test our model using two hidden layers in the neural network. Then after extracting feature from selected character image we train our model using MLP module. We can specify the number of neurons in the hidden layer by giving it as an input in the entry section of our application. If we enter the number of neurons in each hidden layer by separating them using a comma it will be considered that our neural network will have two hidden layers. Then after pressing the Train & Test button our model will be trained and the model will be tested using the csv file which we created by extracting features from the selected image. Then after training & testing the selected character image is predicted.

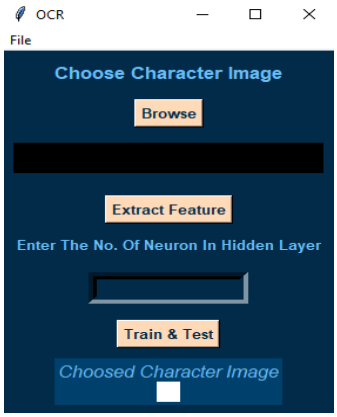


Fig. 11. Our GUI Application

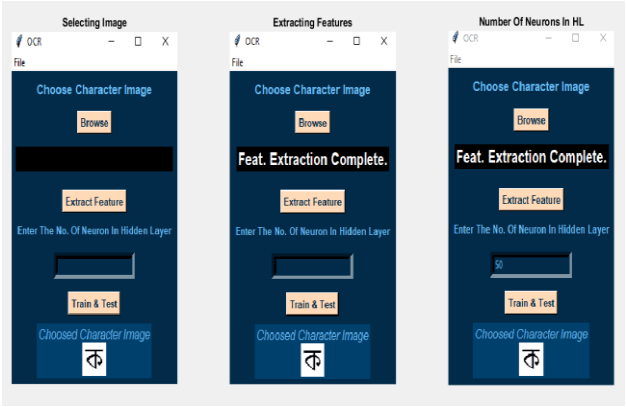


Fig. 12. Our GUI Application

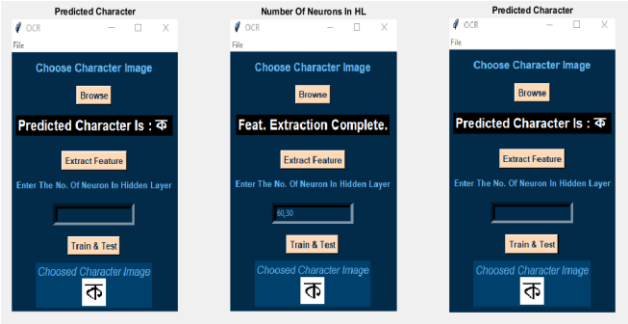


Fig. 13. Our GUI Application

**VIDEO LINK OF THE APPLICATION** : *CLICK HERE*

## VI. Conclusion

In our research we have used different approaches to build our dataset. When we use completely different font for testing the accuracy decreases significantly. So, in order to increase the accuracy, we need to apply some preprocessing steps. We can apply some morphological operation like thinning, erosion and closing etc. When building our dataset we didn't think of different lighting conditions. So, in order to detect character images in different lighting condition with higher accuracy we need to apply histogram equalization on the character images of our dataset. In order to increase accuracy some feature engineering method can also be helpful.

Though we have tried to do our work but there are some things left undone. We think some more features can be implemented

(1) The accuracy for all the letters of Bangla is not same. Recognizing some of the letters can be hard. So, for complex letters some extra feature extraction method is needed so that the letters can be recognized with more accuracy.
(2) We also want to test our dataset with hand written characters to check the accuracy for the hand written character. Our dataset consists of printed characters but we want to try if it can recognize hand written characters.
(3) We want to combine hand written and printed characters in a single dataset. We think a single dataset for both hand written and printed characters should be done so one dataset can be used for both.

## References

[1] Raghuraj Singh1, C. S. Yadav2, Prabhat Verma3,Vibhash Yadav4, Optical Character Recognition (OCR) for Printed Devnagari Script Using Artificial Neural Network International Journal of Computer Science Communication,Vol. 1, No. 1, January- June 2010

[2] Dr.Mrs.V.V.Patil 1, Rajharsh Vishnu Sanap 2 , Rohini Babanrao Kharate,Optical Character Recognition Using Artificial Neural Network,International Journal of Engineering Research and General Science, Volume 3, Issue 1, January-February, 2015

[3] V. Kalaichelvi,Ahammed Shamir Ali, Application of Neural Networks in Character Recognition International Journal of Computer Applications (0975 8887), Volume 52 No.12, August 2012

[4] Shyla Afroge, Boshir Ahmed, Firoz Mahmud, Optical Character Recognition using Back Propagation Neural Network 2nd International Conference on Electrical, Computer Telecommunication Engineering (ICECTE), 8-10 December 2016

[5] Md. Shahiduzzaman, Bangla Hand Written Character Recognition International Journal of Science and Research (IJSR), Index Copernicus Value (2013): 6.14, 2013

[6] Riasat Azim,Wahidur Rahman, M. Fazlul Karim, Bangla Hand Written Character Recognition Using Support Vector Machine. International Journal of Engineering Works, Vol. 3, Issue 6, PP. 36-46, June 2016

[7] Chirag I Patel, Ripal Patel, Palak Patel, Handwritten Character Recognition using Neural Network. International Journal of Scientific Engineering Research, Volume 2, Issue 5,May-2011

[8] Adnan SHatil,Mumit Khan, Minimally Segmenting High Performance Bangla Optical Character Recognition Using Kohonen Network

[9] Md. Mahbub Alam and Dr. M. Abul Kashem, A Complete Bangla OCR System for Printed Chracters COPYRIGHT 2010 JCIT, ISSN 2078-5828 (PRINT), ISSN 2218-5224 (ONLINE),VOLUME 01, ISSUE 01, 2010

[10] Abdur Rahim Md. Forkan, Shuvabrata Saha, Md. Mahfuzur Rahman, Md. Abdus Sattar, RECOGNITION OF CONJUNCTIVE BANGLA CHARACTERS BY ARTIFICIAL NEURAL NETWORK International Conference on Information and Communication Technology, 7-9 March 2007

## Authors' Profiles

**Mr. Asif Isthiaq** has Bachelors degree in Computer Science and Engineering from Ahsanullah University of Science and Technology. Currently he is working as a Software Engineer in mPower Social.

**Ms. Najoa Asreen Saif** is a fresh graduate and has completed Bachelors degree in Computer Science and Engineering from Ahsanullah University of Science and Technology.

**How to cite this paper:** Asif Isthiaq, Najoa Asreen Saif, " OCR for Printed Bangla Characters Using Neural Network", International Journal of Modern Education and Computer Science(IJMECS), Vol.12, No.2, pp. 19-29, 2020.DOI: 10.5815/ijmecs.2020.02.03

Copyright © 2020 MECS 