Bengali Handwritten Character Recognition Using Convolutional Neural Network From Live And Real-world Images

Submitted By Aniket Bhunia And Bishwajeet Ghosh
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

Introduction

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.

Objective

This project seeks to identify individual handwritten Bengali character. We used three different approaches to take inputs. 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. .

Dataset

We used CMATERdb 3.1.2 dataset is developed by Jadavpur University, Kolkata. It has 15000 images of character. The images were 32 x 32px 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 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: Sample image from dataset

Preprocessing of Data

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.
(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. One-hot encoding for the categories to compare in the classification stage.
(4) 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.

Model

The model type that we will be using is Sequential. It allows you to build a model layer by layer. 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) Rectified Linear Activation. 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 'Adaptive Moment Estimation (Adam) ' as our optimizer and 'categorical_crossentropy' for our loss function. 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.

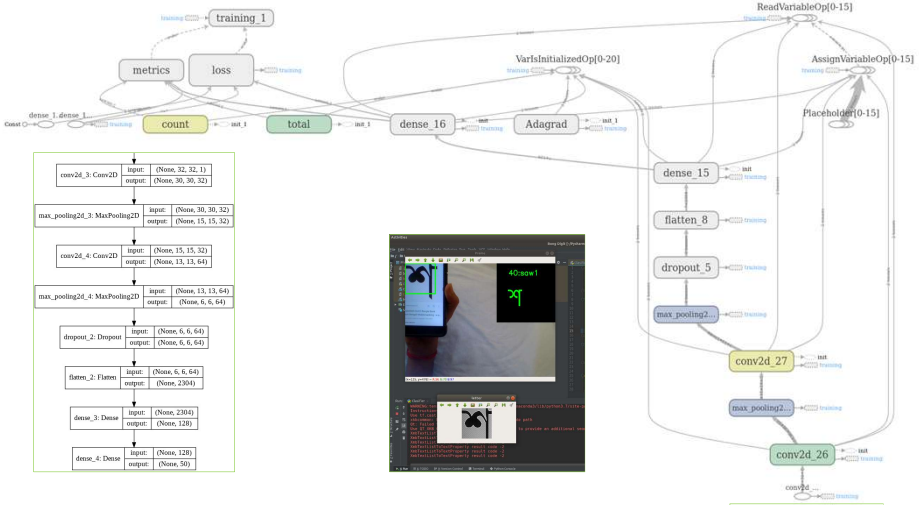

Fig: CNN Diagram

Fig: Screenshots Of Applications

Fig: CNN Workflow

Training & Testing

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.

Image Segmentation

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. 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 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.

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. 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.

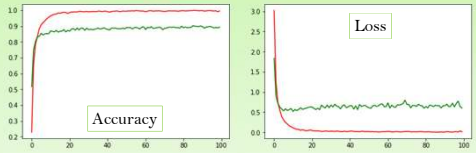

Fig: Graph of Validation and training accuracy and loss

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. 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. We have used python programming language(3.6) and libraries such as 'keras' (to build CNN), 'tensorflow' as backend for training and 'theano' as backend in testing.

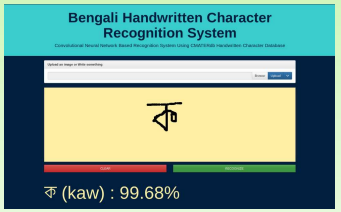

Fig: Screenshots Of Applications

Conclusion and Future Work

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(89.26%) than any other model to the best of my knowledge. 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 We would like to do these things to explore the word recognition problem in Bengali language.

Contact details: aniketbhunia007@gmail.com