Machine Learning Engineer Nanodegree

HANDWRITTEN TELUGU CHARACTER RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS

Harathi Surya Patchipala
December 3, 2017

I. Definition

Project Overview

Neural Networks are recently being used in various kinds of pattern recognition. Handwritings of different persons are different; therefore, it is very difficult to recognize the handwritten characters. Handwritten Character recognition is an area of pattern recognition that has become the subject of research during the last some decades. Neural network is playing an important role in handwritten character recognition. Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch screens and other devices. It can be online or offline. In this context, online recognition involves conversion of digital pen-tip movements into a list of coordinates, used as input for the classification system whereas offline recognition uses images of characters as input.

Although a lot of work has been reported for handwriting recognition in English and Asian languages such as Japanese, Chinese etc., and very few attempts on Indian languages like Hindi, Tamil, Telugu, Kannada etc. In this paper, I am developing handwritten character recognition algorithm for Telugu [South Indian language] with high recognition accuracy and minimum training and classification time.

Some of the earlier works apply shallow learning with hand-designed features on both online and offline datasets. Examples of hand-designed features include pixel densities over regions of image, character curvature, dimensions, and number of horizontal and vertical lines. C. Vikram et.al., used multilayer perceptrons [MLP] and obtained an accuracy of 85% on handwritten Telugu character dataset.

Problem Statement

Handwriting recognition has been one of the active and challenging research areas in the field of image processing and pattern recognition. Since 1929, number of character recognition systems have been developed and are used for even commercial purpose also. Several applications including mail sorting, bank cheques processing, reading aid for blind, document reading and postal address recognition require offline handwriting systems. Working in Postal service need us to decode and deliver something like 30 million handwritten envelopes every single day. The challenges are to do mail-sorting that ensure all those millions of letters reach their destinations.

Character recognition complexity varies among different languages due to distinct shapes, strokes and number of characters. [1]Telugu, a South Indian language that ranks third by the number of native speakers in India. Fifteenth in the Ethnologue list of most-spoken languages worldwide and is the most widely spoken Dravidian language in the world. About 800 million people use Telugu as their speaking and writing purpose. Telugu script has 18 vowels and 36 consonants, of which 13 vowels and 35 consonants are in common usage. Of all the Indic scripts, the Telugu script has the largest number of vowels and consonants. Moreover, Telugu contains many similar shaped characters; in some cases a character differ from its similar one with a full-zero (*anusvāra*) (◌ం), half-zero (*arthanusvāra* or *candrabindu*) (◌ఁ) and *visarga* (◌ః) to convey various shades of nasal sounds. That makes difficult to achieve better performance with simple technique as well as hinders to work with Telugu handwritten character recognition.

Handwritten character recognition can be online or offline. In this context, online recognition involves conversion of digital pen-tip movements into a list of coordinates, used as input for the classification system whereas offline recognition uses images of characters as input. Some of the earlier works apply shallow learning with hand-designed features on both online and offline datasets. Examples of hand-designed features include pixel densities over regions of image, character curvature, dimensions, and number of horizontal and vertical lines.

Metrics

The evaluation metric for the model will be transcription accuracy on the test images in the HP dataset of Telugu characters. Accuracy will be defined as correctly predicting handwritten Telugu character in the image. The model must correctly predict not only the number of digits present

1[1]https://en.wikipedia.org/wiki/Telugu_language but also correctly identify each of those digits. As noted above regarding benchmarking against previous work results, I will evaluate it against some other models like handwritten Hindi/ Tamil/ Bengali character recognition in both accuracy and speed.

II. Analysis

Data Exploration

The dataset is downloaded from HP Labs India website [1]. This dataset contains approx. 270 samples of each of 138 Telugu "characters" written by native Telugu writers. The dataset contains wide variation of distinct characters because of different peoples' writing styles. The characters are made available for download as TIFF files. Some of these character images are very complex shaped and closely correlated with others.

Telugu script has 18 vowels and 36 consonants of which 13 vowels and 35 consonants are in common usage. Telugu script is generally non-cursive in style and hence pen-up usually separates the basic graphemes though not always. So, the basic graphemes of the script i.e. independent vowels, consonants, vowel diacritics and consonant modifiers are included in the symbol set. Also included are some consonant-vowel units which cannot be easily segmented. In addition, the symbol set also contains some symbols which do not have linguistic interpretation but have stable pattern across writers and help reduce the total number of symbols to be collected. So, totally there are 138 symbols. These are all assigned to Unicode characters.

The characters are made available for download as TIFF files. The original unequally sized rectangular images are resized into 80 x 80 square images and saved them as JPG files.

The figure 1 shows the vowels of Telugu Language characters.

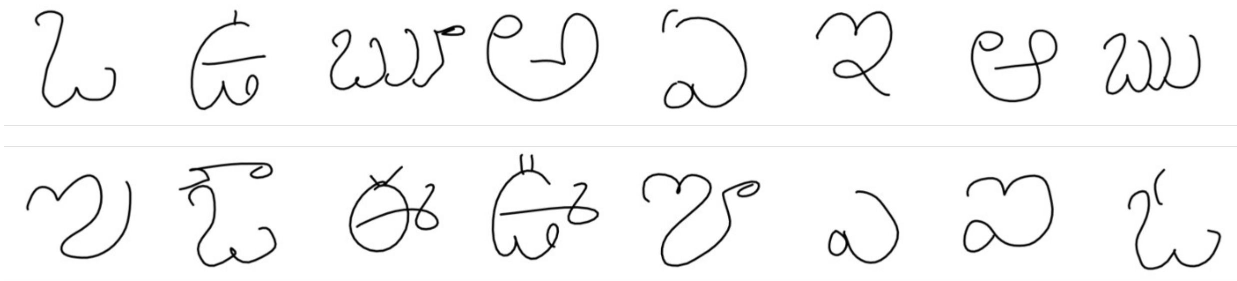

Fig 1: Vowels of Telugu Language

Algorithms and Techniques

Convolutional neural network:

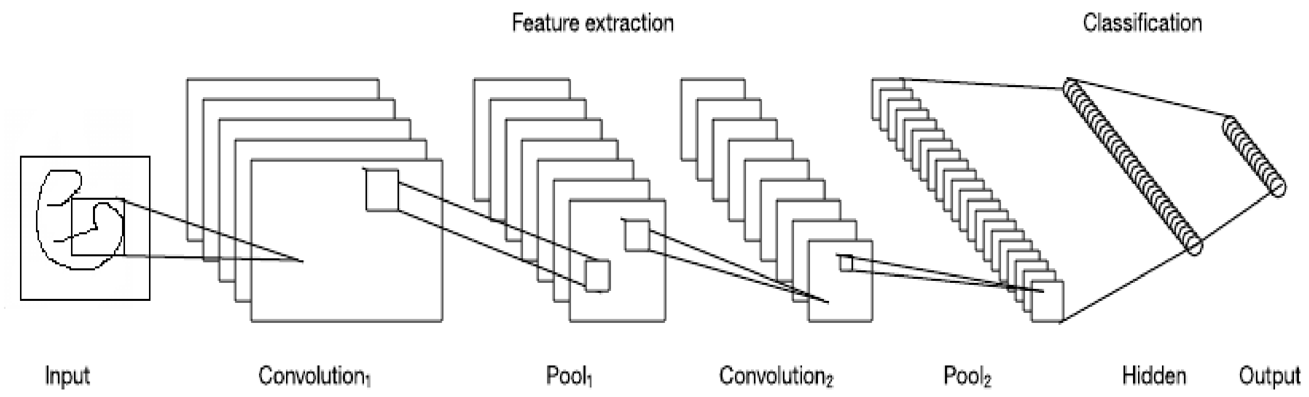

Figure 2: CNN Architecture

Figure 2 shows an overall architecture of CNN that consists with two main parts: feature extraction and classification. In the feature extraction layers, each layer of the network receives the output from its immediate previous layer as its input, and passes the current output as input to the next layer. The CNN architecture is composed with the combination of three types of layers: convolution, max-pooling, and classification. Convolutional layer and max-pooling layer are two types of layers in the low and middle-level of the network. The even numbered layers work for convolution and odd numbered layers work for max-pooling operation. The output nodes of the convolution and maxpooling layers are grouped in to a 2D plane which is called feature mapping. Each plane of the layer usually derived with the combination of one or more planes of the previous layers. The node of the plane is connected to a small region of each connected planes of the previous layer. Each node of the convolution layer extracts features from the input images by convolution operation on the input nodes. The max-pooling layer abstracts features through average or propagating operation on the input nodes.

The higher level features are derived from the propagated feature of the lower level layers. As the features propagate to the highest layer, the dimension of the features is reduced depending on the size of the convolutional and max-pooling masks. However, the number of feature mapping usually increased for mapping the extreme suitable features of the input images to achieve better classification accuracy. The outputs of the last feature maps of CNN are used as input to the fully connected network which is called classification layer. In the classification layer, the desired number of features can be obtained using feature selection techniques depending on the dimension of the weight matrix of the final neural network, then the selected features are set to the classifier to compute confidence of the input images. Based on the highest confidence, the classifier gives outputs for the corresponding classes that the input images belong to. Mathematical details of different layers of CNN are discussed in the following section.

CONVOLUTION LAYER:

In this layer, the feature maps of the previous layer are convolved with learnable kernels such as (Gaussian or Gabor). The outputs of the kernel go through linear or non-linear activation functions such as (sigmoid, hyperbolic tangent, softmax, rectified linear, and identity functions) to form the output feature maps. In general, it can be mathematically modeled as

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} k_{ij}^l + b_j^l\right)$$

where x_j^l is the outputs of the current layer, x_i^{l-1} is previous layer outputs, k_{ij}^l is kernel for present layer, and b_j^l is the bias for current layer. M_j represents a selection of input maps. For each output map is given an additive bias *b*. However, the input maps will be convolved with distinct kernels to generate the corresponding output maps. For instant, the output maps of *j* and *k* both are summation over the input *i* which is in particular applied to the *j*th kernel over the input *i* and takes the summation of its and same operation are being considered for *k*th kernel as well.

3.2.2. SUBSAMPLING LAYER

The subsampling layer performs downsampling operation on the input maps. In this layer, the input and output maps do not change. For example, if there are *N* input maps, then there will be exactly *N* output maps. Due to the downsampling operation, the size of the output maps will be reduced depending on the size of the downsampling mask. In this experiment, 2 × 2 downsampling mask is used. This operation can be formulated as

$$x_j^l = f\left(\beta_j^l down(x_j^{l-1}) + b_j^l\right)$$

where *down*(·) represents a subsampling function. This function usually sums up over *n* × *n* block of the maps from the previous layers and selects the average value or the highest values among the *n* × *n* block maps. Accordingly, the output map dimension is reduced to *n* times with respect to both dimensions of the feature maps. The output maps finally go through linear or non-linear activation functions.

3.2.3. CLASSIFICATION LAYER

This is a fully connected layer which computes the score for each class of the objects using the extracted features from convolutional layer. In this work, the size of the feature map is considered to be 3 x 3 and a feed-forward neural net is used for classification. As for the activation function, softmax function is employed as suggested in most literatures.

Benchmark

I plan to compare the results with the previous works. I will compare the accuracy/mean-squared error to see which is more effective, as well as compare the speed of my method using CDNN and the previous used techniques. Shanthi et al. [4] use pixel densities over different zones of the image as features for an SVM classifier. Their system achieved a recognition rate of 82.04% on a handwritten Tamil character database. K. Mohana Lakshmi et al. [3] achieved a recognition rate of 87.5% on Telugu character dataset using HOG features and Bayesian classification.

Here I want to try to get accuracy greater than 90%. For this I am applying Convolutional Neural Networks which are more efficient than the methods used by the above authors.

III. Methodology

Data Preprocessing

The dataset contains images of contains approx 270 samples of each of 166 Telugu characters written by native Telugu writers. These images are in TIFF format and are unequally sized. The images were under the folders with the names of the writer and the image file name as its Unicode character.

Here I changed the image filename as the writer name followed by the Unicode character so that each image will get a unique file name. And then all the images with the same Unicode were grouped under a folder with Unicode as its folder name so that I can use load_files method in sklearn.datasets[2] module. Then, the original unequally sized rectangular images are resized into 80 x 80 square images and saved them as JPG files.

The preprocessing script is available with this report as preprocess.py.

Implementation

2 http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_files.html#sklearn-datasets-load-files

As mentioned in the Algorithms and Techniques, the classifier used to train the data is Deep Convolutional Neural Network.

The input to the convolutional neural network is a 4D Tensor. The 80 x 80 image is reformatted into a 4D tensor with shape (samples, channels, rows, columns) which is then passed through a stack of different kinds of layers as follows:

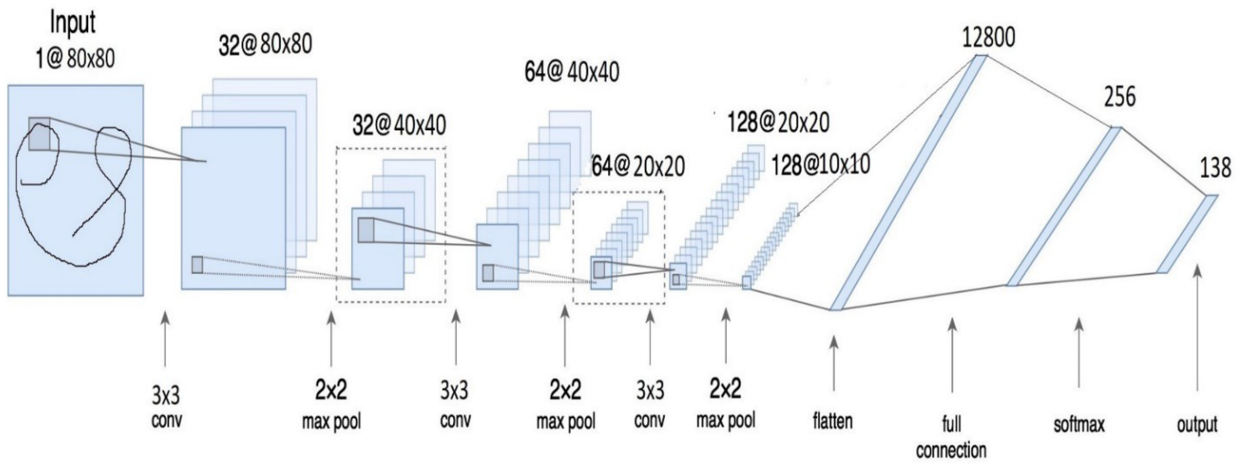

Fig 3 : CNN Architecture for Handwritten Telugu Character Recognition

Input ----------- 4D tensor with shape (n, 80, 80, 1) where n is the number of input images and the number of channels is 1 as the images are binary. (Number of channels = 3 if the images are RGB).

Layer 1 --------- Convolutional Layer with 32 filters and filter size 3 x3 and Max pooling layer with filter size 2 x 2.

Layer 2 --------- Convolutional Layer with 64 filters and filter size 3 x 3 and Max pooling layer with filter size 2 x 2.

Layer 3 --------- Convolutional Layer with 128 filters and filter size 3 x 3 and Max pooling layer with filter size 2 x 2.

Flatten( ) -------- It converts the output of the convolutional part of the CNN into a 1dimensional feature vector, to be used by the fully connected layer.

Layer 4 --------- Fully Connected layer (Dense) with 256 neurons, with dropout regularization rate of 0.4 (probability of 0.4 that any given element will be dropped during training).

Layer 5 --------- Fully Connected layer (Dense) with 138 neurons.

Here, linear activation RELU is used for the convolution and maxpooling layers and SOFTMAX activation is used for the output layer (Fully Connected Layer).

Refinement

Initially I tried the images with size 32x32. When I run the algorithm, I got very less accuracy, then I started changing the optimizer. First I used adam and tried for many combinations of learning rate and decay. Then I tried adadelta and tried for parameter tuning, even though the performance didn't improve.

Then I went through the dataset (i.e., images) which were not clear with size 32 x 32. Then I changed image size to 64 and then 80. Then I noticed some repeated images with different label name which causing the algorithm to get very low performance. Then I removed the similar images with different label names. Then I got 138 characters whereas initially I have 166 characters when I downloaded the data from HP labs website.

I implemented and experimented with the following generalization techniques:

- Dropout in a convolutional layer
- Dropout in a fully connected layer
- Data augmentation function
- Batch Normalization

Applying these techniques did not provide a big boost in performance.

Then with 3 convolutional layers and adam as optimizer, I got 88% test accuracy and 91% training accuracy. But, after changing optimizer to SGD with learning rate 0.1, and adding one more convolutional layer, training accuracy increased to 94% and the test accuracy increased to 91%.

IV. Results

Model Evaluation and Validation

I got test accuracy of 91% and training accuracy of 95% on Telugu character dataset with 20 epochs, if we increase number of epochs, then the accuracy will increase further. The loss reduced from 4.03 to 0.4 as the training progressed. For the first few epochs, the training accuracy is less than the validation accuracy and then after some epochs, train accuracy increased. The validation curves were given in figure 4.

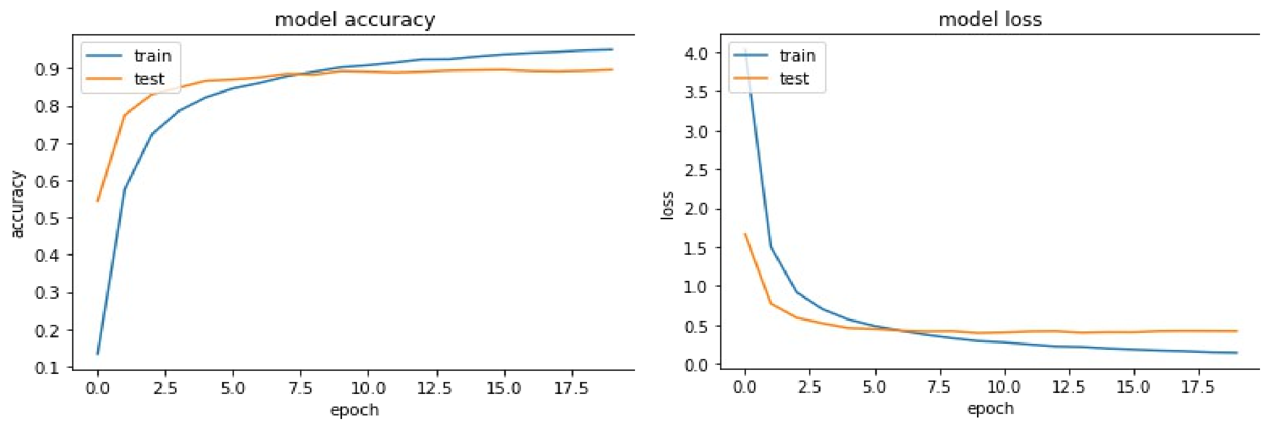

Fig 4 : Validation curves

Justification

The accuracy obtained by the model is greater than the benchmark reported earlier. It's likely that adding more epochs could increase the accuracy further.

The benchmark reported is 90% of test accuracy. But by running 20 epochs, the model obtained 91% accuracy and 95% training accuracy. By adding few more epochs, the accuracy may increase further.

The following table shows the comparison between the accuracy obtained by the CNN model and other models used to classify handwritten Telugu characters.

Method	Accuracy
CNN (Proposed Method)	91%
[1]Zoning Features	78%
[2]Adaptive and Static zoning methods	88.8% for 50 characters
[3]Two Dimensional Fast Fourier Transform and Support Vector Machine	71%
[4]Multi Layer Perceptrons (MLP)	85%
[5]Bayesian Classifier	87.5%

Table 1 : Comparison of different methods and obtained accuracies

V. Conclusion

Free-Form Visualization

The following table shows the actual labels and the predicted labels of the first 50 images which shows the model is predicting almost correct labels except 2 or 3 out of 50 images.

Actual_labels	Predicted_labels
47	47
110	110
46	41
134	134
60	60
67	67
68	68
59	59
121	121
65	65
15	15
119	119
119	119
72	72
44	44
45	45
34	34
126	128
55	55
124	124
21	21
128	128
6	6
121	121
42	42
24	24
107	107
135	135
26	26
28	28
34	34
18	18
44	44
83	58
37	37
44	44
86	86
67	67
85	85
104	104
68	68
22	22
33	33
63	63
124	124
5	98
91	91
30	30
31	31
62	62

Table 2 : Actual labels and the Predicted labels

Confusion Matrix is used to check the performance of the neural network classification. The figure 5 shows the plot of confusion matrix for the model used to classify handwritten Telugu Characters.

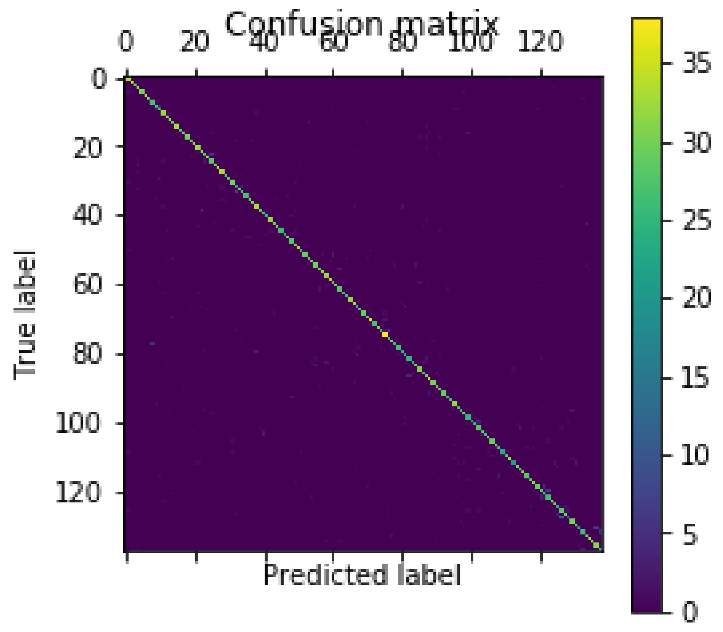

Fig 5 : Confusion Matrix Plot

Reflection

It is well known that the deep convolutional neural networks are very good at classifying image data. There were many experiments done on handwritten character recognition using convolutional neural networks for English alphabets, numbers, Chinese, Arabic and some of the Indian languages like Hindi, Devanagari script etc., But there is very less contribution on Telugu language character recognition. Due to very less contribution, the data available on internet is not so good. I didn’t notice that initially and faced many problems with repeated data. I found that the problem was in data, then I deleted the repeated ones and made the data clear.

Then I found difficult in tuning the algorithm. I tried different optimizers with different learning rates and tried a model changing number of layers and number of filters and filter size.

Finally, I ended up with a model that gives 91% accuracy which exceeded my expectations to get around 85 to 90%.

Improvement

This algorithm used only single characters of Telugu language. But there are many characters with '*vattu*' and '*gunintham*' which were shown in the below figure. As there are very less contributions on Telugu language characters, I couldn't find the dataset consisting of all these characters. But the characters used in this project are like vowels and consonants in English. Due to time constraints I didn't generated dataset with all those characters. In future, I want to extend this algorithm from character recognition to text recognition by creating my own dataset.

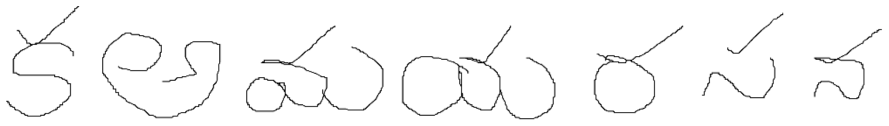

Fig : Some of the characters used in this project

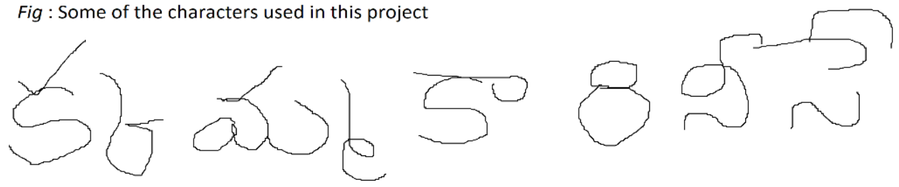

Fig: Examples of the characters that can also be used

References:

[1] P. N. Sastry, T. R. V. Lakshmi, N. V. K. Rao, T. V. Rajinikanth and A. Wahab, "Telugu Handwritten Character Recognition Using Zoning Features," *2014 International Conference on IT Convergence and Security (ICITCS)*, Beijing, 2014.

[2] S. D. Prasad and Y. Kanduri, "Telugu handwritten character recognition using adaptive and static zoning methods," *2016 IEEE Students' Technology Symposium (TechSym)*, Kharagpur, 2016.

[3] Raju Dara, Urmila Panduga, "Telugu Handwritten Isolated Characters Recognition using Two Dimensional Fast Fourier Transform and Support Vector Machine", 2015 International Journal of Computer Applications.

[4] C. Vikram, C. Shoba Bindu, C. Sasikala, "Handwritten Character Recognition for Telugu Scripts Using Multi Layer Perceptrons (MLP)", 2013 International Journal of Advanced Research in Computer Engineering & Technology (IJARCET).

[5] K.Mohana Lakshmi1 ,K.Venkatesh2 ,G.Sunaina3 , D.Sravani4 , P.Dayakar5, Hand Written Telugu Character Recognition Using Bayesian Classifier

Prashanth Vijayaraghavan, Misha Sra, "Handwritten Tamil Recognition using a Convolutional Neural Network". https://web.media.mit.edu/~sra/tamil_cnn.pdf

https://arxiv.org/pdf/1705.02680.pdf

M. Kumar, M. K. Jindal and R. K. Sharma, “Classification of Characters and Grading Writers in Offline Handwritten Gurmukhi Script”, Proceedings of the 2011 International Conference on Image Information Processing, IEEE, pp. 1- 4, (2011).