Digitization of Tamil text

Peratchi Hariharasudhan Kannan*, Vinayak Ravi†
Uni ID: *U6366102, †U6561524

***Abstract*—Tamil characters recognition is extensive and tedious even for human eyes, mainly because of its writing style and its implications of a minor shift-change character relationship in hand-written strokes. Thereby, when Tamil hand-written characters are subjected for classification, it was extremely difficult to optimize, using various machine learning algorithms. Hence classification of characters in Tamil, was performed by using Convolutional Neural Networks (ConvNets) into 149 classes from the HPL dataset, consisting of 49676 characters. The learning was performed by inputting various feature map sizes 8, 16, 32 and 64, in using batch normalization, ReLu activation function, Maxpooling and cross-channel layer, with 4 epochs, under two operating conditions, including two optimizers (Adam & Stochastic Gradient Descent, SGD) or one alone, which was to check the networks robustness. The result for one optimizer (Adam), is test accuracy of 67% and validation accuracy of 59%, whereas for both optimizers, the test accuracy 57% and validation accuracy of 59%. Thereby, showing the robustness for the second type, to be used for various character -recognition learning scenarios, with various datasets.**

I. Introduction

TAMIL language is deemed to be one of the oldest languages in the world, originating as early as 5th century BCE, spoken by many sub-continental countries such as India, Sri Lanka and recognized by Malaysia, Singapore & Indonesia. Pattern recognition of Tamil characters, are essentially very difficult, owing to the large category of scripts (which includes 31 letters in the independent form and 216 combinational letters). The problem arises in identifying the hand-written combinational characters, where a vowel marker is added on to the consonant, even to human eye. Various machine learning methodologies were proposed in character recognition, however our focus is on using Convolutional Neural Network (CNN), utilizing local response normalization, max pooling, with varying feature maps size and learning rates, to classify the Tamil characters that had been inputted.

II. Literature Review

Character recognition originated in 1870s, where Fourier dAlbe found Optophone[1], which was aid given to blind, which scans texts and generates time-varying chords of tones, in identifying letters [1]. Later, extent of character identification developed onto, Optimal Character Recognition (OCR) was used in recognizing characters, which is a process using computer to scan digitized textual input, into machine encoded data. This process facilitated one of the primary proponents in achieving essential interface between man and machine, which automated processes in identifying necessary intel, in accordance to character recognition.

As rule of thumb, pattern recognition was identified to be split into two as template based and feature based approach. Template based approach, uses an ideal template and superimposes the input character, and analysis on correlation between both are performed, which outputs classification, which is a methodology ideally followed by earlier OCR-based methods. Recent studies couple both feature extraction and template matching, to achieve optimized results [2]. The Feature-based approach are of two-types spatial-domain (which derives pixels from the inputted image) and transform-domain (where the inputted image is first transferred into another space, using Fourier Cosine, Wavelet transform and then used to derive the features). Further, Graph-based, moment-based methodologies have also tested for object recognition, which will receive raw/ normalized pattern during the training phase and the system will minimize the miscalculation error, where the primary example is Artificial Neural Network (ANN), which adjusts the weights on each link during the training session.

In considering the various methods, Siromoney et.al [3] utilized, character string dictionary-based method, which deploys row and column wise scanning from character-input matrix, which enables encoding of inputted machine printed Tamil characters. Further [4], used labelled graphs methodology, which used the inputted Tamil characters as primitives (line-like elements) and converted them to labelled graph, which necessarily estimated the correlation coefficients, with labelled-data corresponding to the input characters. Additionally[5], researched on using Fourier descriptors with Neural Networks in recognition, thereby training their network being inputted hand-written Tamil characters, by both male and female participants.

Consecutively [6], argued that the usage of Fourier features in the network descriptors provided diminishing results and hence recommended in usage of wavelet features, to enhance class separability. In regards to that [6] analyzed, combination of local features (using pre-processed (x,y) coordinates) and global features (using Discrete Fourier Transform and Discrete Cosine Transform) , thereby training and testing using Support Vector Machine (SVM) and Radial Bias Filter (RBF) on the IWFHR Tamil Training Dataset. Additionally [7], explored on the idea of converting the inputted hand-written image into Unicode characters by, segmenting the input image into paragraphs using spatial detection technique, converting those paragraphs into lines using vertical histograms and the lines into words and glyphs using horizontal histograms. The extracted features are inputted into self-organizing maps, Fuzzy neural networks, thereby aiding in character recognition. Further [8] used, Convolutional Neural Networks (ConvNets), by varying various pooling (max, stochastic), activation functions (tanh, ReLu), Classifiers (SVM, Softmax), coupled with local response normalization in classifying, the Tamil characters. The scope of our project is to train and classify Tamil

characters from the HPL dataset , using Convolutional Neural Network, essentially linking the input characters with its corresponding labels, by deciphering nuances in hand-written character strokes.

III. Architecture

The input to the network is images of size (32 x 32x1), followed by convolutional (Conv2D) layer with varying feature map sizes of [8,16,32,64] which is stacked in ascending order with filter size of 5 x 5, and corresponding layers after each iteration of feature maps size [8,16,32,64] are followed by, a batch normalization layer, a ReLu Layer, max pooling of size 2 x 2 with stride of 2 and a cross channel layer. Upon completion of the corresponding layers for a specific feature map size, the same is repeated for varying batch sizes and upon completion of the feature map (size 64), it is followed by a fully connected layer classifying 149 classes of the dataset, 149-way softmax classification layer and classification layer. We implemented, rectified linear unit (ReLu) for the Conv2d layers, which is a function $f(x) = max(0, x)$, which returns 0 if it receives any negative input and the value itself if it receives x value.

A. Convolution 2D

Conv2d network takes in the input image and assign weights & biases to the objects of the images (Gray-scale) from HPL database. The primary difference between the feed-forward neural network and the CNN is the pre-processing of images, flattening them, instead CNN helps in successfully capturing spatial and temporal dependencies in an input image. It helps in learning the sophistication of image, by ensuring reusability of weights. The output of a CNN layer, ensures in maintenance of features of the grayscale images character input, thereby improving the prediction accuracy. The (5 x 5 Filter) used helps in performing the convolution operation, by moving the filer across the image, with a stride of 2, thereby performing matrix multiplication operation between filter k and portion p of the image. The filter will have the same depth as that of the convolutional layer, with W- width of image and H height of the image, with Fw = width and Fh = height, of the filter, sw and sw as, horizontal and vertical stride of convolution, where P is the zero padding provided (in our case no zero padding) [9].

$$output\ width = \frac{W - F_w + 2P}{S_w} + 1 \tag{1}$$

$$output\ height = \frac{W - F_h + 2P}{S_h} + 1 \tag{2}$$

The Conv2D network for varying feature size, extracts and generates various feature maps, which acts as a low-medium level feature extraction in the network upon end of complete iteration. The Layer is inputted with two optimizers (Adam & Stochastic Gradient Descent, SGD) and made robust in operations and was examined for both the cases of its individualistic performance, pertaining to the overall networks performance. SGD maintains a single learning rate for weight updating and learning rate doesnt change much, which is expressed as equation 3 [10]:

procedure BatchNormalisation($minibatch$)
$y_i = [BN]_{\gamma,\beta}(x_i)$
$\mu_\beta \leftarrow \frac{1}{m}\Sigma_{i=1}^{m}[x_i]$ ▷ mini-batch mean
$\sigma_\beta^2 \leftarrow \frac{1}{m}\Sigma_{i=1}^{m}[x_i - \mu_\beta]^2]$ ▷ mean-batch variance
$\hat{x_i} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$ ▷ normalize
$y_i \leftarrow \gamma\hat{x_i} + \beta \equiv BN_{\gamma,\beta}\ x_i$ ▷ Scale and shift
end procedure

$$Q(w) = \frac{1}{n}\Sigma_{i=1}^{n}Q_i(w) \tag{3}$$

Where, parameter w is used to minimize Q(w), which is to be estimated. Whereas [10], Adam optimizer computes the individual learning rates using adaption, for the different parameters foe the first and second moment of gradients. The reason for choosing Adam, is because it combines the incentives of both Adaptive Gradient Algorithm (AdaGrad), which utilizes per-parameter learning rate, thereby improving cases linked with sparse gradients and also Root Mean Square Propagation (RMSProp), which also uses the per-parameter learning rates, which aids in tackling the magnitude shift for gradient weights and also for noise-oriented parameters.The Learning rate and validation accuracy for the optimizers are shown as in Table I.

B. Batch Normalization Layer

This layer is used in adjustment and scaling the activations. Normalizing is done to ramp up the reaction rate, thereby enhancing training speed [11]. Batch Normalization aids in covariance shift, which is necessarily, change in distribution of input variables in the training/ test data. This will act as an instigator or minimizer in all the other modules network learning rate. It also aids in reduction of overfitting, as it possesses slight regularization effects, and hence usage of dropouts in the network is reduced, thereby aiding in retainment of information. The normalized output is multiplied by standard deviation and mean parameter, which necessarily lets Stochastic Gradient Descent (SGD) do denormalization by changing only the two weights, instead of losing the stability of the network, with change in weights. The algorithm for applying batch normalization transform is shown as given below:

The Rectified Linear Unit (ReLU) Layer helps in identifying the features extracted by the batch normalization layer which concentrates the features of an image from convolution layer. The ReLU layer helps in reducing the negative gradient descent created by convolution layer[10]. Post-ReLU Layer pooling is done to down sample on the data before sending it to the next convolution/ fully connected layer.

IV. Data Collection and Processing

The data collection process for the task involved lots of literature review. The open source data was collected from HP India Pvt. Ltd., which contained 155 different letters of

Table I: Comparison of Parameters and Accuracy

Trials	Parameters and Accuracy		
	Optimizer	***Learning Rate***	***Validation Accuracy***
1	SGDM (without data augmentation)	0.01	51%
2	ADAM (without data augmentation)	0.01	57%
3	ADAM (with data augmentation)	0.01	59%
4	SGDM (with data augmentation)	0.01	53%

the language with a minimum of 167 in each letter. For improvement, the data collected was reshaped with uniform dimensions of 32 x 32 x 1. Furthermore due to inconsistency of the amount of images in each class, a few classes were neglected. Finally, 149 classes were chosen to for the final training and evaluation. A separate piece of code [7] was written in order to bring the uniform dimensions into the data set. The data set has about 50683 images from 155 classes of which 6 classes were neglected due to insufficient data and the final data set had 49676 images from 149 classes. All images are in gray scale, which can be further augmented by using various rectification techniques, in accordance to proper correlation of the data, overcoming skewed handwritten pattern, erratic hand movements and the inaccuracies in the scanning methodology utilized.

Certain noises and inaccuracies can be bettered by utilizing methods suggested by [2], which consists of 'thresholding' which is a task of binarizing the image, into black and white, thereby separating the foreground infromation from background. Secondly, 'skeletonization' is performed, which is essentially thinning, the output image from thresholding, into thin lines,i.e. refining pixels by removal, without affecting its original structure. Finally, 'normalization' is done to convert varying sized images, which minimizes within class variances, into one standard size of images (32 x 32 images), by doing so it will aid in maximizing classification accuracy.

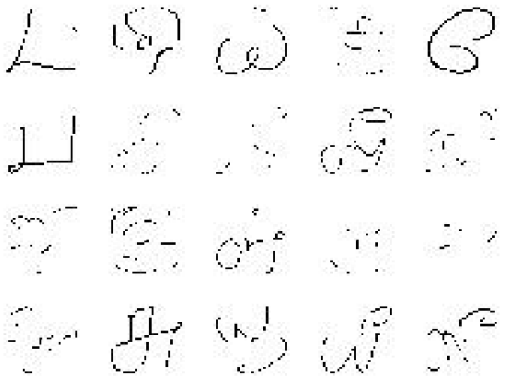

Figure 1: Sample data

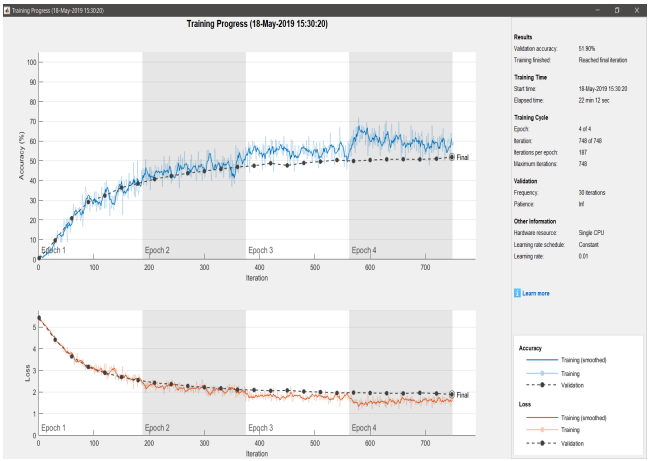

Figure 2: Initial training and loss graph

V. Network

A. Structure

The network was constructed by studying regular MNIST classification, image input of size 32 x 32 is sent into the network, where the batch was normalized using a *batch normalization layer*. Normally a batch normalization layer would normalize the activation of a particular convolution layer with respect to mini batch mean and standard deviation. In our network, we use cross channel normalization, after the convolution layers where it normalizes with respect to channel wise mean than the whole mini batch. The fully connected layer is connected among its classes and then to a softmax and classification layer outputting the class of a particular image.

B. Training

The nature of convolution neural network is that it provides variety of combination of parameters which can be for the good or at times bad for the network training. With the reduction of mini batch size (32 and 64) we realized that the training time reduced significantly compared to the batch size of 128.

As it is clear from the above image post-saturation of validation accuracy, the network is basically over fitting the data, in order to avoid that we change the mini batch size and continue our training. Further, the training was carried out with varying important parameters like learning rate, optimizers keeping the number of epochs constant with each change in parameters while training.

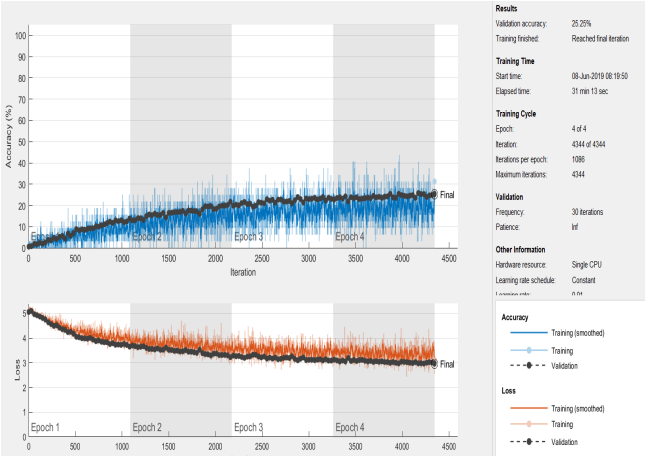

Figure 3: Change in mini batch size from 128 to 32

The data augmentation is one of the important techniques to improve the variations in data and to get more generalized network which can detect variety of data. The figure shows the accuracy of the network when the input is augmented data.

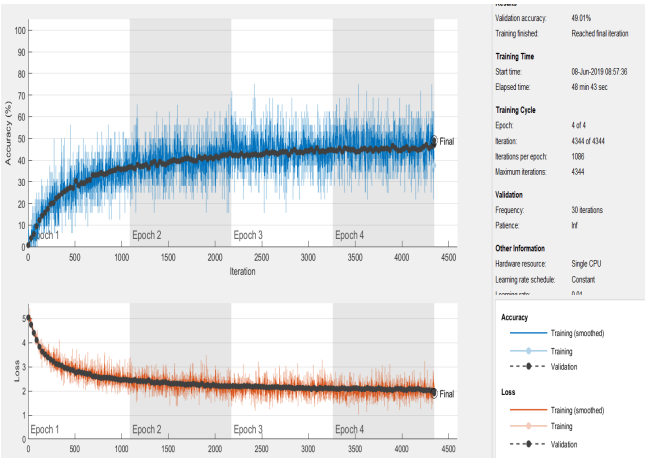

Figure 4: Training without data augmentation

VI. Code snippets

```
[imdsTrain,imdsTest,imdsVal] =
     splitEachLabel(imds, 0.7, 0.15,
     0.15, 'randomized');
imageInputLayer([32 32 1], "Name",
    "imageinput")

convolution2dLayer([5 5], 8, "Name",
    "conv_1", "Padding", "same",
    "WeightLearnRateFactor",1)

reluLayer("Name","relu_1")

batchNormalizationLayer

maxPooling2dLayer([2 2], "Name","
    maxpool_1", "Padding",
    "same", "Stride", [2 2])
crossChannelNormalizationLayer(5, "Name",
    "crossnorm_1", "Alpha", 0.1)
```

The above snippets show a brief overview of the dataset split up into training set (70 %), validation set (15 %) and test set (15 %). Further, inputting the snippet shows the inputting of normalized images of size (32 x 32 x 1) and initial layers of the network, consisting of conv2D layer (filter size 5 x 5), a ReLu layer, a batch normalization layer and a max pooling layer (with filter size 2 x 2 and stride 2).

VII. Results and further improvements

The following images shows the feature maps of few convolution layers of the network. These feature maps gives the network on how the network detects the features of images that are sent into layers of the network.The network was evaluated and made robust by continuously training and testing varying the hyper parameters. The validation accuracy grazed over 59% and test accuracy was about 66%. With robust model, the validation accuracy was 59% and test accuracy 56%.

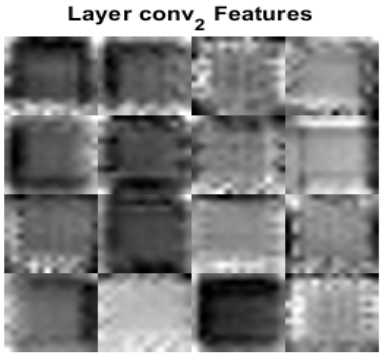

Figure 5: Feature map of convolution layer 2

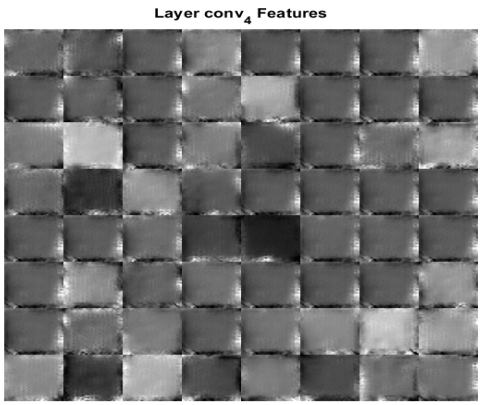

Figure 6: Feature map of convolution layer 4

VIII. Conclusion

The proposed works well with the given data and its translations. As a further improvement the data set can be expanded for the general character recognition. Due to the nature of the combination of letters in the language it is complicated to make a computer understand the scripting patterns. In the view of further improvement, it can be implemented in small, test or prototype scale. In order to recognize earlier/ old scripts of the languages, it is necessary to understand the transformation of pattern over time. These techniques can be updated and extended for handwritten text recognition of the language with proper threshold and segmentation. CNN proved to be one of the aids in developing a robust character recognizer, with which multiple augmentation can be performed, to even better the performance and increase the character recognition accuracy.

References

[1] *En.wikipedia.org*. 2019. URL: http://en.wikipedia.org/.

[2] N. Shanthi and K. Duraiswamy. "A novel svm-based handwritten tamil character recognition system". In: *Pattern Analysis and Applications* 13 (2 2010), pp. 173–180.

[3] R. Chandrasekaran G. Siromoney and M. Chandrasekaran. "Computer recognition of printed Tamil character". In: *Pattern Recognit* 10 (1978), pp. 243–247.

[4] P. Chinnuswamy and S.G. Krishnamoorthy. "Recognition of hand printed Tamil characters". In: *Pattern Recognit* 12 (2 1980), pp. 141–152.

[5] J. Sutha and N.Ramaraj. "Neural network based ofine Tamil handwritten character recognition system". In: *Proceedings of the international conference on computational intelligence and multimedia applications* 2 (2 2007), pp. 446–450.

[6] A. G. Ramakrishnan and K. B. Urala. "Global and local features for recognition of online handwritten numerals and tamil characters". In: *Proceedings of the 4th International Workshop on Multilingual OCR, MOCR* 12 (2 1980), pp. 1–16.

[7] C. Sureshkumar and T. Ravichandran. "Handwritten tamil character recognition and conversion using neural network". In: *IntJComputSciEng* 2 (7 2010), pp. 2261–2267.

[8] P. Vijayaraghavan and M. Sra. "Handwritten Tamil Recognition using a Convolutional Neural Network". In: ().

[9] *Batch normalization in Neural Networks*. 2019. URL: towardsdatascience.com.

[10] J. Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. 2019. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[11] *Stochastic gradient descent*. 2019. URL: en.wikipedia.org.