Handwritten Tamil vowel character recognition using convolutional neural network

Saravanan Govindarajan
Vellore Institute of Technology
Chennai
saravanan.2016@vitstudent.ac.in

Vikrame Vasudev
Vellore Institute of Technology
Vellore
vikrame1999@gmail.com

***Abstract*— Handwritten Character Recognition (HCR) is an active and challenging area of research . It is heavily used in various real-time applications. It plays an important role in the conversion of handwritten information from images to machine editable texts. This paper describes the use of a deep learning model, Convolutional Neural Networks (CNN) to recognize handwritten Tamil vowel from images. About 4500 image samples of the vowel characters were used to train the model. The size of the input image used is 24 x 24 pixels. The proposed model achieved 98% accuracy on training images and 96% accuracy on test images after 30 training iterations.**

Keywords— Character Recognition, CNN, Deep Learning, HCR

I. INTRODUCTION

Character recognition from images has a great potential in many applications such as scanning zip codes, invoice imaging, number plates processing etc. Character recognition faces challenging problems due to the possibilities of variations in the size, styles, and orientation of the characters. In order to achieve higher accuracy, it is necessary to have a well-defined feature extraction procedure along with a good classifier. Over the years, various approaches have been proposed by different researchers for handwritten character recognition of various languages like Chinese [1], Arabic [2], English [3] etc. Many frameworks have been developed for recognition of various Indian languages but not much work has been done towards the Tamil Language.

Tamil is a south Indian language and one of the oldest languages in the world. It is also one of the most widely used languages in the world. Alphabet set of Tamil language consists of 12 vowels, 18 consonants, 1 special character (ak) and 216 composite alphabets obtained by combining the vowels and consonants. This paper focuses on recognising handwritten Tamil vowels. The list of Tamil vowels are shown in Fig.1.

Previous approaches used in recognition of Tamil characters include template-matching [4] and hand coding of features [5]. Convolutional neural networks have the ability to learn the features from pixels and also to classify each of those pixels.

Fig.1. Vowel characters in Tamil alphabet set

Deep learning models such as convolutional neural networks (CNNs) prove to be state-of-the-art models for image pattern recognition tasks due to its ability to extract complex features and provide mappings from the given images. They were initially recognized for their performance in the object recognition task in the IMAGENET competition [6]. The main problem in pattern recognition models is that they are significantly sensitive to both translational and scale variance of the features in the images but CNN models are significantly insensitive to both, which makes these models more suitable for real world images.

In our work, we studied the application of convolutional neural networks to recognize the handwritten Tamil vowel characters and also analysed the effect of model parameters such as number of layers and learning rate on the prediction accuracy. The rest of the paper is organised as follows, Section II presents the image datasets used for training and evaluating the model. Section III presents the image pre-processing techniques. Section IV introduces the methodology behind the convolutional neural network. Section V and Section VI present the proposed model architecture and the experimental results. Conclusion is given in Section VII.

978-1-5386-5630-3/18/$31.00 ©2018 IEEE

II. DATASET

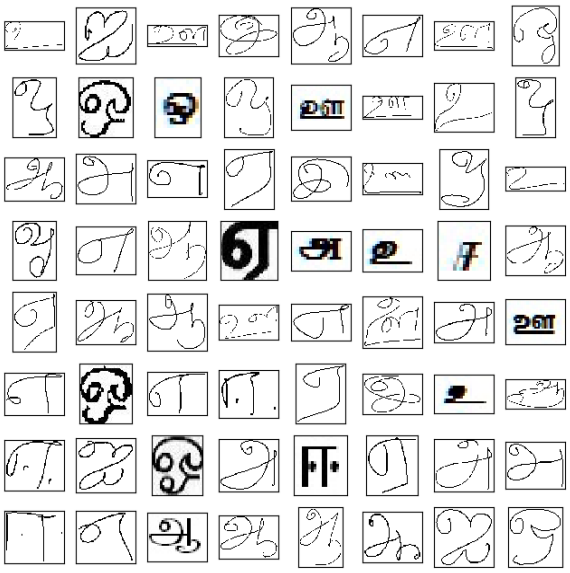

Fig.2. Image samples from the collected dataset

IWFHR 10 dataset [7] and UJTDchar dataset [8] are used for training the model. The IWFHR 10 dataset contains approximately 270 labelled image samples in TIFF format for each vowel character, which was collected by HP India Labs. The UJTDchar dataset contains 100 labelled image samples in JPG format for each character. 4500 images of handwritten Tamil vowel characters were collected after merging these two datasets. The collected images were composed of different sizes, styles and of different intensities as shown in Fig. 2.

III. IMAGE PREPROCESSING

The image dataset was split into training images (3150 images) and test images (1350 images). The dataset constitutes images of different sizes and ranges of pixel values, the images were pre-processed to make them uniform throughout. The following steps were applied to all training and test images.

1. Resizing the image to 24 x 24 pixels.
2. Normalising the image pixels values to range between 0 and 1. This is carried out by dividing all the pixel values of the image by 255.

All the image pre-processing was done using open source library OpenCV of Python [9].

IV. CONVOLUTIONAL NEURAL NETWORK

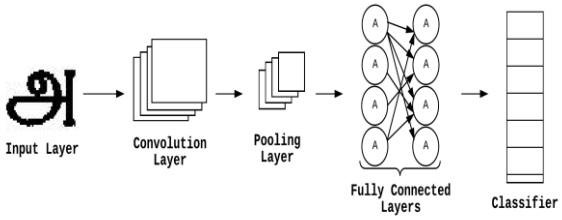

Fig. 3. Overview of a convolutional neural network

Convolutional neural networks (CNNs) are primarily supervised deep neural networks used for classification of images, object recognitions and image segmentation applications [10]. More recently, CNNs are also applied to applications involving sound, texts [11][12].

CNNs are successful and efficient models because of their ability to recognize complex patterns present in images without any hard coding of details or features. The CNN model uses the concept of receptive field and weight sharing, which not only reduces the training parameters but in turn reduces the complexity of the overall model. The feature map of each layer is generated from the previous layer's receptive field which makes it more suitable for pattern recognition. Normally, a CNN model consists of an input layer, followed by a few alternating layers of convolution, activation, pooling and then followed by the fully connected network layers at the very end of the neural network. These models are trained using the back-propagation algorithm combined with an optimiser such as Stochastic Gradient Descent algorithm or Adam optimizer. Overview of a single layer CNN model is shown is Fig. 3. The architecture of the proposed CNN model is presented in Section V.

A. Convolution Layer

Convolution layer extracts the tiles of the input image and applies filters to them to compute convolved features. The convolved feature map may have different size and depth than the input tile. Two main parameters define this layer, the size of the tiles that are extracted and the depth of the output feature map. The depth of the output feature map is equal to the number of filters that are applied. Majority of the CNN's computations occur in this layer.

B. Activation Layer

To implement complex function mappings from input to output, non-linearities are introduced into the network in the form of activation functions. The commonly used activation functions are Rectified Linear Unit (ReLU) [13], softmax, sigmoid and tanh. For this network, ReLU as in (1) is used as an activation function for all the layers except the final layer in this network. The final layer uses the softmax activation function as in (2). The softmax function calculates the probabilities of each target classes, the class with highest probability is chosen as the predicted class.

ReLU Activation function,

$$f(x) = max(0, x) \tag{1}$$

where *x* is the output of previous layer.

Softmax activation function,

$$f(x_i) = \frac{\mathrm{Exp}(x_i)}{\sum_{j=0}^{n} \mathrm{Exp}(x_j)} \tag{2}$$

where i = 0, 1, 2, … n and x_i is i^{th} sample of the extracted feature vector and $f(x_i)$ is corresponding probability.

C. Pooling Layer

This layer downsamples the convolved feature, reducing the number of dimensions of the extracted feature map, while still preserving the feature's information. Two parameters define this operation, pool size and stride length. The pooling layer slides a window of pool size and moves it over stride length pixels over the feature map and extracts new feature maps of specified size. Max pooling is used in this network. The maximum value of each feature map is output to the new feature map.

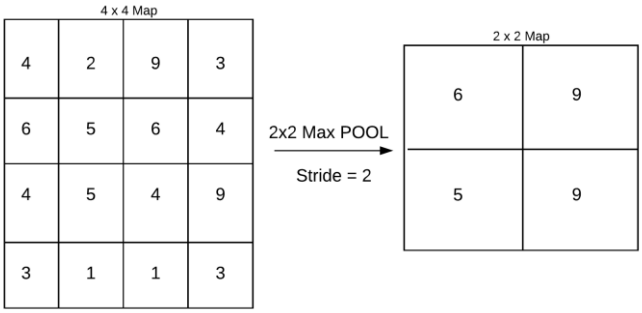

Fig.4. Max Pooling Operation

D. Batch Normalization Layer

Batch normalization is the technique of normalizing the outputs in each hidden layer according to (3). Normalization is done in such a way that the mean of output activation is zero and standard deviation is one. This technique improves the performance and stability of the network [14]. Given a batch of N examples to the batch normalization layer, each of which is a D-dimensional vector, and is represented by the matrix $X \in \mathbb{R}^{NxD}$, where each row x_i is a single example and is normalised by (3).

$$x_i \rightarrow \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{3}$$

where, μ and σ^2 are the mean and variance of each input dimension across the batch. ϵ is a small constant to prevent division by zero. The μ and σ^2 are computed by (4) and (5),

$$\mu = \frac{1}{N}\sum_i x_i \tag{4}$$

$$\sigma^2 = \frac{1}{N}\sum_i (x_i - \mu)^2 \tag{5}$$

E. Dropout

Network units are dropped randomly from the network during training. This prevents units from co-adapting too much. Dropout is done to prevent the convolutional neural network from overfitting as illustrated in [15].

F. Fully Connected Layer

Fully Connected layers are present at the end of the convolutional neural network. This layer performs the actual classification based on the feature tiles extracted by the convolutional layers. For this network, the final fully connected layer uses softmax activation function as in (2), to obtain a probability value between 0 and 1 for each of the classes the network is trying to predict.

This layer basically takes an input volume and outputs an N dimensional vector, where N is the number of output classes. In this network, N corresponds to 12 (12 vowels). Each number in this N dimensional vector corresponds to probability of a certain class.

V. Implementation of Proposed CNN

Fig. 6 shows the illustration of the proposed network using Tensorboard, an open source tool from Tensorflow [17] to visualize the network's flow and Fig. 5 illustrates the architecture of the proposed network. The proposed network consists of 2 convolutional layers, 2 max pooling layers and fully connected layers at the end for performing the image classification. Also, batch normalisation and dropout layers are implemented in between layers to prevent the network from overfitting and to increase the stability of the network.

Layer (type)	Output Shape	Param #
CONVOLUTION_LAYER1 (Conv2D)	(None, 24, 24, 32)	896
RELU1 (Activation)	(None, 24, 24, 32)	0
BATCH_NORM1 (BatchNormalizat	(None, 24, 24, 32)	96
MAX_POOL1 (MaxPooling2D)	(None, 12, 12, 32)	0
CONVOLUTION_LAYER2 (Conv2D)	(None, 12, 12, 64)	18496
RELU2 (Activation)	(None, 12, 12, 64)	0
BATCH_NORM2 (BatchNormalizat	(None, 12, 12, 64)	48
MAX_POOL2 (MaxPooling2D)	(None, 6, 6, 64)	0
DROPOUT (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
FC1 (Dense)	(None, 1024)	2360320
RELU3 (Activation)	(None, 1024)	0
BATCH_NORM3 (BatchNormalizat	(None, 1024)	4096
FC2 (Dense)	(None, 12)	12300
SOFTMAX (Activation)	(None, 12)	0

Total params: 2,396,252
Trainable params: 2,394,132
Non-trainable params: 2,120

Fig. 5. Architecture of the proposed Network

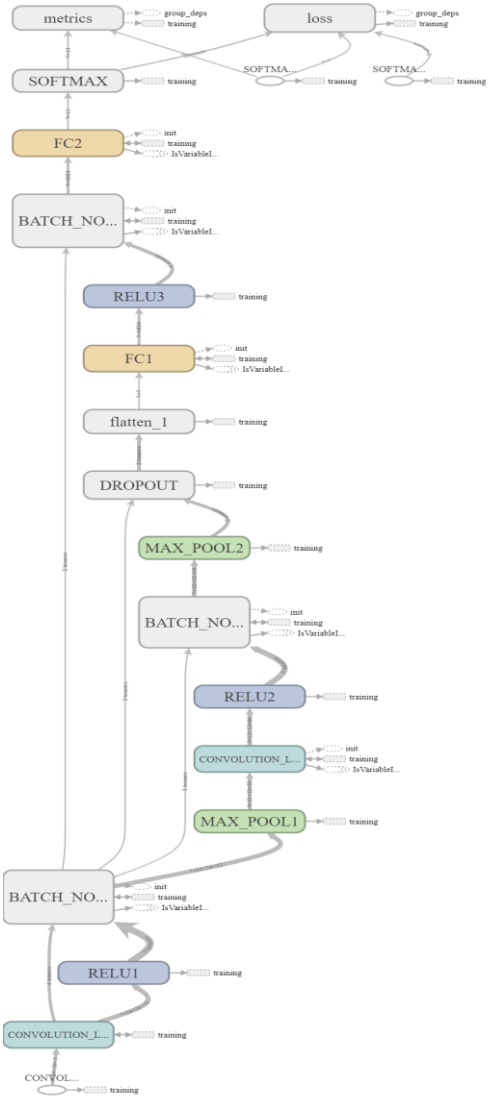

Fig. 6. Tensorboard visualization of the proposed model

The input of the network is 24 x 24 x 3 sized images, these images pass through first convolutional layer consisting of 3 x 3 sized 32 filters bank. Thereby producing 24 x 24 x 32 sized feature maps. These feature maps are then passed through ReLU and batch normalization layers. The output maps of batch normalization layers are then passed through pooling layer with pool size 2 and strides 2, therefore the output maps of pooling layer will be of size 12 x 12 x 32. The pooled maps pass through the second convolutional layer consisting of 3 x 3 sized 64 filters bank. The convolved feature maps then pass through ReLU, batch normalization and max pooling layers producing 6 x 6 x 64 feature maps.

These feature maps act as the input to the fully connected layers where the actual classification occurs. Layer 1 consists of 1024 units which is connected to Layer 2 with 12 units (each unit corresponds to one of the 12 vowels) through connection weights. In order to obtain probabilities of each classes (12 classes overall), softmax activation function is used. All the parameters of this network are tuned using the Adam optimiser [13] with backpropagation algorithm.

VI. RESULT AND DISCUSSION

In order to attain the best possible results, the network's parameters such as learning rate, number of layers have been altered and the changes in accuracies are tabulated in TABLE 1. All the experiments were implemented in Python using Keras [18] library with Tensorflow Backend and were trained on NVIDIA Geforce 1050Ti GPU.

Based on the results from TABLE 1, the training is then performed on 2-layer convolutional neural network by setting the learning rate as 0.001. The model is evaluated using the test images of different styles and fonts. Overall, the trained model was able to achieve 98% accuracy on the training images and 96% accuracy on the test images. Top 3 predictions from the model are shown in Fig.7. for different test images.

TABLE I. EXPERIMENTS WITH LEARNING RATE AND NUMBER OF CONV LAYERS

Number of conv layers	*Epochs*	*Learning Rate*	*Training Set Accuracy*	*Test Set Accuracy*
1	15	0.01	91%	84%
1	15	0.001	93%	86%
2	15	0.01	88%	78%
2	15	0.001	96%	92%
1	30	0.01	94%	88%
1	30	0.001	95%	91%
2	30	0.01	94%	90%
2	30	0.001	98%	96%

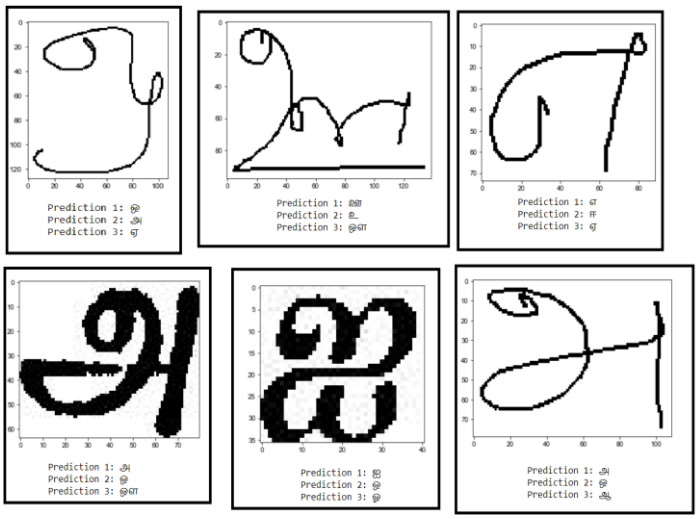

Fig. 7. Predictions

VII. Conclusion

In, this paper we studied the application of a deep learning model, convolutional neural network to perform handwritten Tamil vowel character recognition. A 2-layer convolutional neural network model was implemented to recognize the characters. The proposed model was trained and tested using real world image samples of handwritten Tamil vowel characters collected from IFWHR 10 dataset and UJTDchar dataset. The total number of image samples used in this experiment is 4500 images for 12 classes. The model achieved an accuracy of 96 % on the test images with 30 iterations of training. Furthermore, the effect of number of layers and the learning rate on the accuracy were also studied in this paper.

References

[1] Liu, Cheng-Lin, et al. "Online and offline handwritten Chinese character recognition: benchmarking on new databases." Pattern Recognition 46.1 (2013): 155-162.

[2] Lawgali, Ahmed. "A survey on arabic character recognition." International Journal of Signal Processing, Image Processing and Pattern Recognition 8.2 (2015): 401-426.

[3] Patil, Vijay, and Sanjay Shimpi. "Handwritten English character recognition using neural network." Elixir Comput Sci Eng 41 (2011): 5587-5591.

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

[5]] C. Sureshkumar and T. Ravichandran. Handwritten tamil character recognition and conversion using neural network. Int J Comput Sci Eng, 2(7):2261–67, 2010.

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[7] IFWHR 10 , http://lipitk.sourceforge.net/datasets/tamilchardata.htm

[8] UJTDchar dataset, http://www.jfn.ac.lk/index.php/data-sets-printed-tamil-characters-printed-documents/

[9] Bradski, Gary, and Adrian Kaehler. "OpenCV." *Dr. Dobb's journal of software tools* (2000).

[10] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[11] Salamon, Justin, and Juan Pablo Bello. "Deep convolutional neural networks and data augmentation for environmental sound classification." *IEEE Signal Processing Letters* 24.3 (2017): 279-283.

[12] dos Santos, Cicero, and Maira Gatti. "Deep convolutional neural networks for sentiment analysis of short texts." *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2014.

[13] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.

[14] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

[15] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[16] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*(2014).

[17] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." *OSDI*. Vol. 16. 2016.

[18] Chollet, François. "Keras." (2015).