Journal of King Saud University – Computer and Information Sciences 34 (2022) 1183–1190

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Benchmarking on offline Handwritten Tamil Character Recognition using convolutional neural networks

Kavitha B.R. [a,*], Srimathi C. [b]

[a] School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India
[b] School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

ARTICLE INFO

Article history:
Received 6 March 2019
Revised 9 May 2019
Accepted 7 June 2019
Available online 15 June 2019

Keywords:
Handwritten Tamil Character Recognition
CNN

ABSTRACT

Convolutional Neural Networks (CNN) are playing a vital role nowadays in every aspect of computer vision applications. In this paper we have used the state of the art CNN in recognizing handwritten Tamil characters in offline mode. CNNs differ from traditional approach of Handwritten Tamil Character Recognition (HTCR) in extracting the features automatically. We have used an isolated handwritten Tamil character dataset developed by HP Labs India. We have developed a CNN model from scratch by training the model with the Tamil characters in offline mode and have achieved good recognition results on both the training and testing datasets. This work is an attempt to set a benchmark for offline HTCR using deep learning techniques. This work have produced a training accuracy of 95.16% which is far better compared to the traditional approaches.

1. Introduction

Tamil is one of the ancient Indian languages which is predominantly used in Southern India, Srilanka and Malaysia. The speciality of Tamil language is every sound pronounced has a syllable in Tamil. The economy of characters to represent a word is minimal in Tamil language. The smallest unit of Tamil script is syllable. These syllabic units of Tamil script has 12 vowels, 18 consonants and a special character (Ayudha Ezhuthu, ஃ). The combination of vowels and consonants (Table 1) make a total of 216 compound characters, thus a total of 247 characters. There are 5 borrowed consonants from Sanskrit, which when combined with vowels would yield another 60 compound characters so as to make a total of 307 characters. The complete Tamil character set (Table 2) can be represented by combinations of 156 distinct characters (Table 3). For example, கௌ (pronounced as *kow*) can be represented by combining ெ','க','ள'.

Handwritten Character Recognition (HCR) is of two forms: online and offline. Online method converts the tip movements (strokes) of digital pen to a list of coordinates, whereas offline method uses characters as scanned images. The challenging part of handwritten character recognition is the variations in the writing patterns of individuals. Even the same person handwriting can vary at different times. Traditional machine learning approaches were used for an offline HCR for a long time. A typical machine learning way of handwritten character recognition would be preprocessing, segmenting, extracting the features and classifying. An offline HCR system is first trained with the set of characters (as scanned images) and later when a new character image is given as input, the system should be able to recognize it accurately. HCR has shown its usefulness in many applications such as sorting of mails in post offices, bank check reading, digitization of legacy documents and legal documents and handwritten document/form conversions.

Digitization of a handwritten document involves various operations such as conversion of color or grayscale image to binary image, extraction of the foreground text from background texture (if any), noise removal, separation of the individual lines, segmentation of words in each line, segmentation of the characters in every word and recognition of the isolated characters. However, this work involves only the isolated offline handwritten Tamil characters recognition using deep learning approach.

Other languages like Chinese, Japanese, Arabic, Hangul have several standard datasets and researchers have set benchmark for the same. Chinese have National Laboratory of Pattern Recognition (NLPR) and Institute of Automation of Chinese Academy

* Corresponding author.
E-mail address: kavitha.br@vit.ac.in (B.R. Kavitha).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2019.06.004
1319-1578/

Table 1
Combination of Vowels and Consonants in Tamil. First column has 12 vowels. Second column represents 18 consonants. Third column is the combination of a consonant with all the vowels resulting in the fourth column which are compound characters. Every consonant is combined with all the vowels to produce 216 compound characters.

Vowels	Consonants	Consonant + vowels	Compound Characters
அ	க்	க்+அ	க
ஆ	ங்	க்+ஆ	கா
இ	ச்	க்+இ	கி
ஈ	ஞ்	க்+ஈ	கீ
உ	ட்	க்+உ	கு
ஊ	ண்	க்+ஊ	கூ
எ	த்	க்+எ	கெ
ஏ	ந்	க்+ஏ	கே
ஐ	ப்	க்+ஐ	கை
ஒ	ம்	க்+ஒ	கொ
ஓ	ய்	க்+ஓ	கோ
ஔ	ர்	க்+ஔ	கௌ
	ல்		
	வ்		
	ழ்		
	ள்		
	ற்		
	ன்		

Sciences (CASIA) for promoting the research activities in Chinese character recognition with datasets such as CASIA-OLHWDB and CASIA-HWDB of 3755 classes. Arabic have datasets OIHADB and AHCD for 28 classes. Japanese have the Electrotechnical Laboratory (ETL) Character Database with the latest dataset ETL9B having 3036 classes. For Tamil, HPLabs, India have developed a dataset named 'hpl-tamil-iso-char' for 156 classes of Tamil characters which was used by very few. In (Vijayaraghavan and Sra, 2014), only 34 classes were used. In (Bhattacharya et al., 2007) though all the classes were used, they used a grouping method. Shanthi and Duraiswamy (2010) created their own dataset and was able to recognize only 34 classes. This work will set the benchmark for HTCR for all the 156 classes using deep learning methods. Chinese and Japanese languages have proven best results with the state of the art CNNs which was the motivation behind this work. The objective of our work is to set a benchmark for Tamil character dataset of HPLabs.

The remainder of this paper is organized as follows. Section 2 discusses about the related work in the literature. Section 3 presents the general convolutional neural network framework. Section 4 explains the proposed architecture for HTCR and steps

Table 2
Complete Tamil Character set.

	தமிழ் மொழி எழுத்துக்கள்(Tamil Characters)												
	உயிர் எழுத்துக்கள்(Vowels)												
	அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ	ஃ
உயிர் மெய் எழுத்துக்கள் (Compound characters)	க	கா	கி	கீ	கு	கூ	கெ	கே	கை	கொ	கோ	கௌ	க்
	ங	ஙா	ஙி	ஙீ	ஙு	ஙூ	ஙெ	ஙே	ஙை	ஙொ	ஙோ	ஙௌ	ங்
	ச	சா	சி	சீ	சு	சூ	செ	சே	சை	சொ	சோ	சௌ	ச்
	ஞ	ஞா	ஞி	ஞீ	ஞு	ஞூ	ஞெ	ஞே	ஞை	ஞொ	ஞோ	ஞௌ	ஞ்
	ட	டா	டி	டீ	டு	டூ	டெ	டே	டை	டொ	டோ	டௌ	ட்
	ண	ணா	ணி	ணீ	ணு	ணூ	ணெ	ணே	ணை	ணொ	ணோ	ணௌ	ண்
	த	தா	தி	தீ	து	தூ	தெ	தே	தை	தொ	தோ	தௌ	த்
	ந	நா	நி	நீ	நு	நூ	நெ	நே	நை	நொ	நோ	நௌ	ந்
	ப	பா	பி	பீ	பு	பூ	பெ	பே	பை	பொ	போ	பௌ	ப்
	ம	மா	மி	மீ	மு	மூ	மெ	மே	மை	மொ	மோ	மௌ	ம்
	ய	யா	யி	யீ	யு	யூ	யெ	யே	யை	யொ	யோ	யௌ	ய்
	ர	ரா	ரி	ரீ	ரு	ரூ	ரெ	ரே	ரை	ரொ	ரோ	ரௌ	ர்
	ல	லா	லி	லீ	லு	லூ	லெ	லே	லை	லொ	லோ	லௌ	ல்
	வ	வா	வி	வீ	வு	வூ	வெ	வே	வை	வொ	வோ	வௌ	வ்
	ழ	ழா	ழி	ழீ	ழு	ழூ	ழெ	ழே	ழை	ழொ	ழோ	ழௌ	ழ்
	ள	ளா	ளி	ளீ	ளு	ளூ	ளெ	ளே	ளை	ளொ	ளோ	ளௌ	ள்
	ற	றா	றி	றீ	று	றூ	றெ	றே	றை	றொ	றோ	றௌ	ற்
	ன	னா	னி	னீ	னு	னூ	னெ	னே	னை	னொ	னோ	னௌ	ன்
	வடமொழி எழுத்துக்கள்(Borrowed Consonants)								சிறப்பு எழுத்து-ஸ்ரீ				
	ஜ	ஜா	ஜி	ஜீ	ஜு	ஜூ	ஜெ	ஜே	ஜை	ஜொ	ஜோ	ஜௌ	ஜ்
	ஷ	ஷா	ஷி	ஷீ	ஷு	ஷூ	ஷெ	ஷே	ஷை	ஷொ	ஷோ	ஷௌ	ஷ்
	ஸ	ஸா	ஸி	ஸீ	ஸு	ஸூ	ஸெ	ஸே	ஸை	ஸொ	ஸோ	ஸௌ	ஸ்
	ஹ	ஹா	ஹி	ஹீ	ஹு	ஹூ	ஹெ	ஹே	ஹை	ஹொ	ஹோ	ஹௌ	ஹ்
	க்ஷ	க்ஷா	க்ஷி	க்ஷீ	க்ஷு	க்ஷூ	க்ஷெ	க்ஷே	க்ஷை	க்ஷொ	க்ஷோ	க்ஷௌ	க்ஷ்

Table 3
The 'hpl-tamil-iso-char' dataset with their class labels.

அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ			
0	1	2	3	4	5	6	7	8	9	10	11			
க	ங	ச	ஞ	ட	ண	த	ந	ப	ம	ய	ர	ல	வ	ழ
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
ள	ற	ன	ஸ	ஷ	ஜ	ஹ	க்ஷ	கி	ஙி	சி	ஞி	டி	ணி	தி
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
நி	பி	மி	யி	ரி	லி	வி	ழி	ளி	றி	னி	ஸி	ஷி	ஜி	ஹி
42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
க்ஷி	கீ	ஙீ	சீ	ஞீ	ட்ீ	ணீ	தீ	நீ	பீ	மீ	யீ	ரீ	லீ	வீ
57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
ழீ	ளீ	றீ	னீ	ஸீ	ஷீ	ஜீ	ஹீ	க்ஷீ	கு	ஙு	சு	ஞு	டு	ணு
72	73	74	75	76	77	78	79	80	81	82	83	84	85	86
து	நு	பு	மு	யு	ரு	லு	வு	ழு	ளு	று	னு	ஷு	ஜு	ஹு
87	88	89	90	91	92	93	94	95	96	97	98	99	100	101
ஞு	டு	ணு	து	நு	பு	மு	யு	ரு	லு	வு	ழு	ஸு	று	னு
102	103	104	105	106	107	108	109	110	111	112	113	114	115	116
ா	ி	ீ	ை	ஸ்ரீ	ஸு	ஷு	ஜு	ஹு	க்ஷு	ஸூ	ஷூ	ஜூ	ஹூ	க்ஷூ
117	118	119	120	121	122	123	124	125	126	127	128	129	130	131
க்	ங்	ச்	ஞ்	ட்	ண்	த்	ந்	ப்	ம்	ய்	ர்	ல்	வ்	ழ்
132	133	134	135	136	137	138	139	140	141	142	143	144	145	146
ள்	ற்	ன்	ஸ்	ஷ்	ஜ்	ஹ்	க்ஷ்	ஃ						
147	148	149	150	151	152	153	154	155						

involved in it. Section 5 describes the dataset used, experimental setup and training and testing procedures. Section 6 presents the result analysis and discussion. The last section gives the conclusion of the paper.

2. Related work

Though deep learning approaches were widely used in Handwritten Character Recognition in many languages such as Chinese, Arabic, English etc., in Tamil most of the work done till date were using traditional approaches. A typical approach for HTCR using traditional machine learning techniques would follow preprocessing, segmentation of characters, feature extraction, classification and then predicting the new characters. Shanthi and Duraiswamy (2010) proposed a model in which features extracted are pixel densities and SVM classifier is used for classification of 106 classes. They achieved an accuracy of 82.04% for 34 characters. Jose and Wahi (2013) used wavelet transform for feature extraction and for classification used a Backpropagation neural network with which have achieved 89% recognition accuracy. Sureshkumar and Ravichandran (2010) have extracted features from each character glyphs with various attributes and classified using Support Vector Machines (SVM), Self Organizing Maps (SOM), Fuzzy network, RCS algorithm and Radial basis function. Bhattacharya et al. (2007) have proposed a two stage recognition method in which an unsupervised clustering is used in first stage for grouping the character classes and a supervised classifier is used in second stage for recognizing the characters, thereby an accuracy of 89.66% was achieved. Using convolutional neural networks, a recent work has been done by Vijayaraghavan and Sra (2014), reporting an accuracy of 94.4% with 35 classes.

Deep convolutional neural networks have been successful in handwritten Chinese character recognition. Ciregan et al. (2012) have tested the Chinese character dataset, CASIA (Institute of Automation of Chinese Academy of Sciences) composed of 300 samples of 3755 characters as one of the experiments using Multi Column Deep Neural Network. Cireşan and Meier (2015) have worked on the same dataset with eight different network architectures and have achieved near human accuracy. Several benchmarked datasets are available for Chinese characters with over million characters (Liu et al., 2013). Zhang et al. (2017) have proposed a new benchmark by combining CNN with normalization, direction decomposed feature map and adaptation layer with which achieved an accuracy of 97.37% on offline task. Tsai (2016) have used CNNs to recognize three types of handwritten Japanese scripts namely, hiragana, katakana and kanji. The overall classification accuracy recorded 99.53% for 1004 classes.

Boufenar et al. (2017) have developed a CNN model from scratch for the Arabic character set, OIHACDB-28 and have achieved 97.32% accuracy. They also presented another work which had used transfer learning and achieved an accuracy of 100% for the model developed from scratch. Earlier, in the same year, El-Sawy et al. (2017) have developed a CNN from scratch with an accuracy of 94.9% and Elleuch et al. (2017) have used Deep Belief Network (DBN) architecture along with the regularization techniques, dropout and dropconnect to produce an error classification rate of 2.73% and 2.27% respectively.

3. Background

CNNs are the widely used deep learning models in handling image related tasks like image recognition, image classification, image captioning etc. These networks are generally a combination of convolution layers, pooling layers and fully connected layers (Fig. 1). These three blocks are used to construct a CNN model by varying the number of blocks, adding or deleting a block. Various architectures have been developed since 2012 Imagenet competition which had reduced the misclassification rate from 15.6% to 3.7% over 4 years (Krizhevsky et al., 2012, Zeiler and Fergus, 2014, Simonyan and Zisserman, 2014, Szegedy et al., 2015, He et al., 2016, Canziani et al., 2016, Russakovsky et al., 2015). Each architecture had varying hyperparameters or had used new methods like Drop out (Srivastava et al., 2014), Batch normalization (Ioffe and Szegedy, 2015) etc.

3.1. Convolution layer

Convolution layer differ with a neural network in a way that, not every pixel (a neuron) is connected to the next layer with a weight and bias, but the entire image is split as small regions (say a *nxn* matrix) and weights and bias are applied over it. These weights and bias are referred to as filters or kernels which when convoluted with every small region in the input image would yield feature maps. These filters are the simple 'features' that is searched

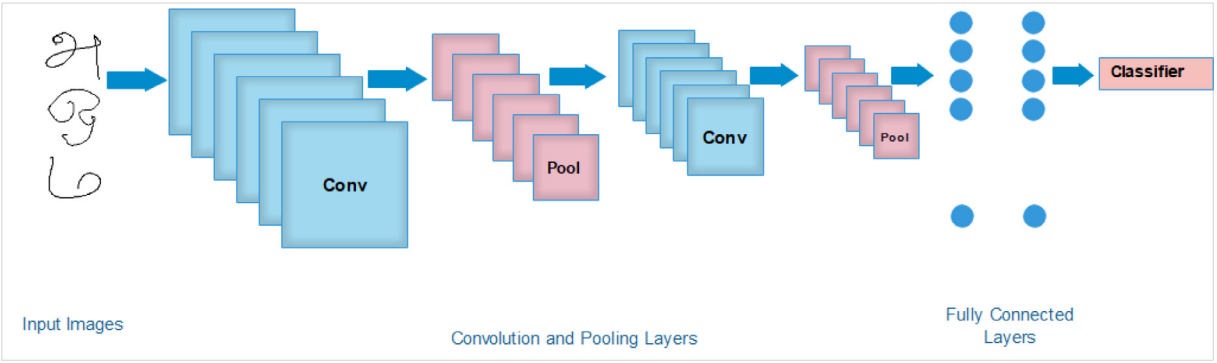

Fig. 1. Convolution Neural Network.

in the input image in the convolution layer. The number of parameters required for this convolution operation would be minimal as the same filter is traversed over the entire image for a single feature. The number of filters, size of the local region, stride, and padding are the hyper parameters of convolution layer. Based on the size and genre of the input image, these hyper parameters could be tuned to achieve better results.

3.2. Pooling layer

In order to reduce the spatial dimension of the image as well as the number of parameters, thereby to reduce the computation, this pooling layer is used. This layer performs a fixed function over the input, hence no parameters are introduced. Different types of pooling are available such as average pooling, stochastic pooling, max pooling. Max pooling is the most commonly used pooling algorithm, in which an *nxn* window is slid across and down the input with a stride value *s* and for each position the maximum value in the *nxn* region is taken, thereby reducing the size of the input. This layer provides translational invariance such that even with a slight variation in the position would still be able to recognize the image. But the location information is lost as the size is reduced.

3.3. Fully connected layer

In this layer the flattened output of the last pooling layer is fed as input to a fully connected layer. This layer behaves like a traditional neural network layer where every neuron of the previous layer is connected to the present layer. Hence, the number of parameters in this layer are higher compared to the convolution layer. This fully connected layer is connected to an output layer which is generally a classifier.

3.4. Activation function

Different activation functions have been used across various architectures of convolution neural networks. Nonlinear activation functions such as ReLU, LReLU, PReLU, and Swish have proven better results when compared to the classic sigmoid or tangent functions. These nonlinear functions have helped in speeding up the training. In this work we have tried different activation functions and found ReLU to be more effective than others.

4. Proposed method

The proposed model consists of 2 parts: training part and recognition part. Training part involves data pre-processing, building the network architecture and training the network with the pre-processed data. The recognition part involves pre-processing of data and recognizing the character using the trained model.

4.1. Pre-processing

The dataset contains 82,929 images in tiff or png format. These images are obtained from the online version using simple piecewise linear interpolation and a constant thickening factor. The images are bi-level images with background being white (255) and the foreground in black (0). The images are of varying sizes

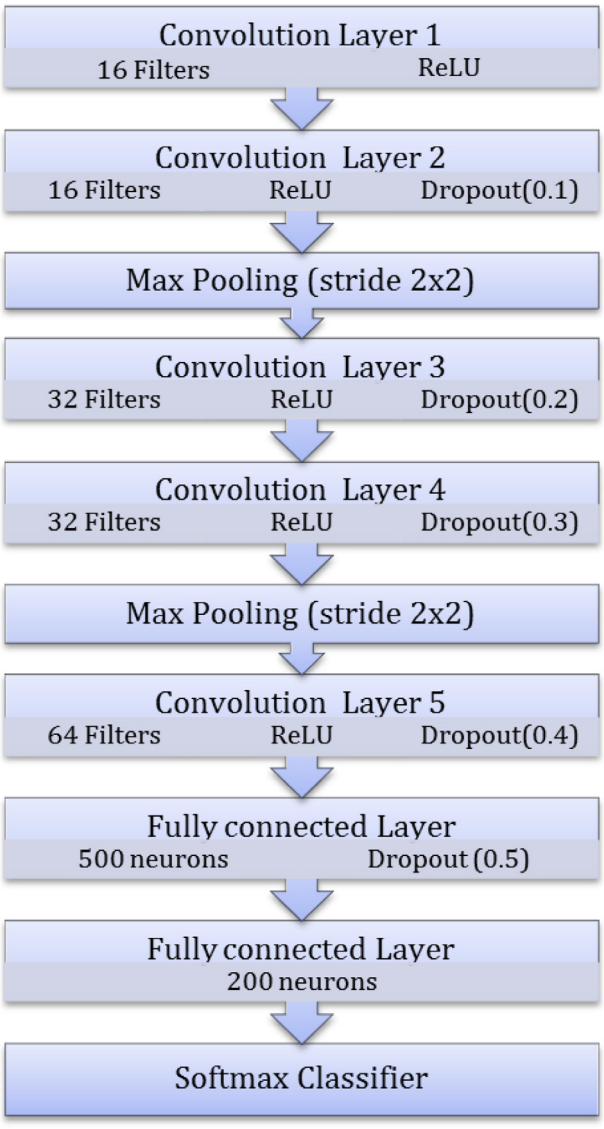

Fig. 2. Architecture of the proposed model.

which were size normalized to 64 × 64 using bilinear interpolation technique and scaled to 0, 1 range. We performed training on two set of inputs, one with the original images and another with inverted images (foreground as 1 and background as 255). However, there was no significant differences in terms of accuracy or training time.

4.2. Architecture

The proposed architecture for HTCR consists of nine layers: five convolutional layers, two max pooling layers and two fully connected layers. The network architecture can be described as follows: 16C3-16C3-MP2-32C3-32C3-MP2-64C3-500N-200N, where nC_i represents a convolution layer with n feature maps and $i \times i$ filters, MP_j represents a max pooling layer with $j \times j$ kernel and kN represents a fully connected layer with k neurons (Fig. 2). Each convolution layer includes an activation function. This model used ReLU, a non-linear activation function. The input layer consists of images of 64x64 size and the output layer is a softmax classifier with 156 classes.

Table 4
Hyper parameters used in our architecture.

Hyper parameters	Values
Initialization	Xavier
Batch Size	64
Optimizer	Adam
Epochs	100
Learning rate	0.001
Activation function	ReLU

4.3. Hyper parameters

Several hyper parameters are required to run the network, some of them are architecture specific while some are needed for the effective training of the network. Different hyper parameters such as stride, padding, and depth are used at each of the layers which could be tuned to build a better model. We have used a stride value of 1 to slide one pixel over the input image in the convolutional layer and pooling layer. We used zero padding to maintain the same input shape in the output such that no information is lost at the borders.

We have used Xavier initialization (Glorot and Bengio, 2010) for weights. We made all the experimental runs up to 30 epochs as the convergence generally happened in 20 to 30 epochs. However, we

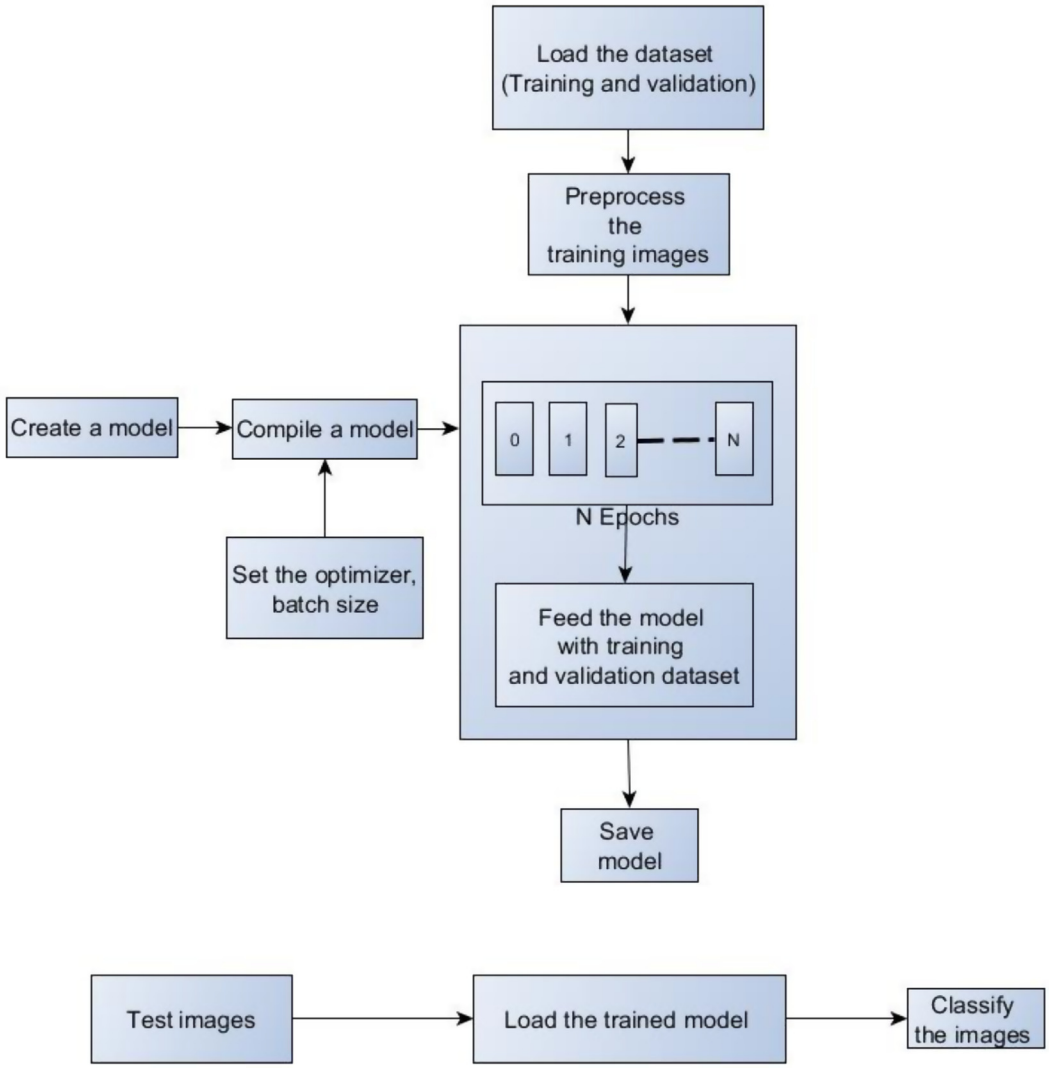

Fig. 3. The complete training and testing process of our system.

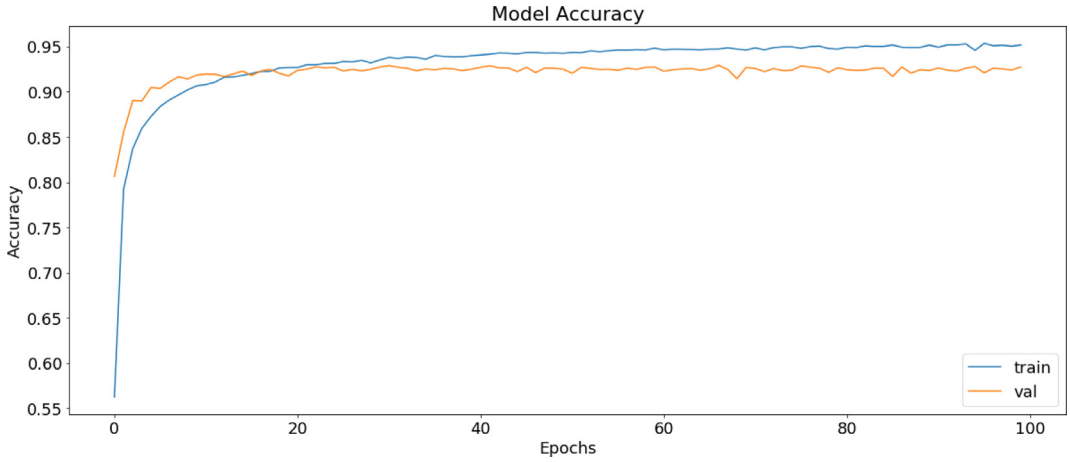

Fig. 4. Training and validation accuracy of the proposed model.

Table 5
Comparison of our approach with previous work.

Authors	Dataset	No. of classes	Method	Training Accuracy (%)	Test Accuracy (%)
Bhattacharya et al. (2007)	HPLabs dataset	156	Clustering and groupwise classification	92.77	89.66
Shanthi and Duraiswamy (2010)	Own dataset	34	SVM	–	82.04
Vijayaraghavan and Sra (2014)	HPLabs dataset	35	CNN	99	94.4
Our work	HPLabs dataset	156	CNN	95.16	97.7

have run the best result model up to 100 epochs. The input images of size 64x64 were fed to the network in batches of size 64. Different batch sizes were experimented, but a nominal size of 64 was used due to memory constraints. Different hyper parameters are presented in Table 4.

5. Experimental setup

5.1. Dataset

This work uses the Isolated Handwritten Tamil Character dataset developed by HP Labs India. This dataset consists of 156 different Tamil characters (hpl-tamil-*iso*-char) written by native Tamil writers from various cities of Southern India using HP TabletPC[1]. The dataset contains approximately 500 samples for each class (with very few classes having around 300 samples) with a total of 82,928 samples and is freely available. The entire Tamil character set can be represented with these 156 unique characters (Table 3).

5.2. Training and testing process

In this experimental setup, all the 156 classes are used, where each class has around 500 samples. A total of 82,928 images were present in the dataset which was split into training and validation images during run time on 80:20 ratio. For training and testing the neural networks, Keras was used as deep learning framework with Tensorflow (Abadi et al., 2016) as backend. The entire training and testing was performed on a Windows 64-bit desktop personal computer with an Intel Core i5 Processor (CPU) and 8 GB RAM. Fig. 3 shows the complete training and testing process of our HTCR system.

A CNN model is created with the architecture specified in previous section. The model is compiled after setting few attributes such as optimizers, batch size. We have used Adam optimizer with a learning rate of 0.001(Table 4). After many trails of different batch sizes, we have fixed batch size to 64. Higher values of batch size could not be run as the entire set up is on a CPU. Once the model is compiled, the pre-processed dataset is loaded to the model. The training was done for 100 epochs, after each epoch validation is done on the validation part. We have used early stopping method to avoid over training of the network. After tuning the parameters for various number of filters, kernel size, the final model with optimum accuracy is saved. The saved model is used for testing a completely new data (test dataset is available in HPLabs dataset) which was not seen in training or validation. The test set consists of 26,926 samples with 170 samples per class and the ground truth is also available.

6. Results and discussion

The baseline architecture of ours gave 99.37% training accuracy and 89% validation accuracy which clearly denotes the overfitting. So we implemented few methods to overcome the same. One such method is dropout, a kind of regularization technique which was introduced in each of the convolution layer present in the network starting with a probability of 0.1 and increasing by 0.1 in each layer (Fig. 2). With dropout, we were able to reduce the overfitting with a trade-off in bias which corresponds to a training accuracy of 95.16% and validation accuracy of 92.74%. The total time taken for training is 37 h in a CPU system with 8 GB RAM. The training and validation accuracy are displayed in Fig. 4. The test set was feed into the trained model which gave a recognition accuracy of 97.7%.

As specified in Table 5, the results are compared with previous work. In the work done by Bhattacharya et al., k-means clustering was used to form groups of character classes and a MLP classifier was used which did classification as whether the character belonged to the group or not. So, we cannot compare these results

[1] http://lipitk.sourceforge.net/datasets/tamilchardata.htm

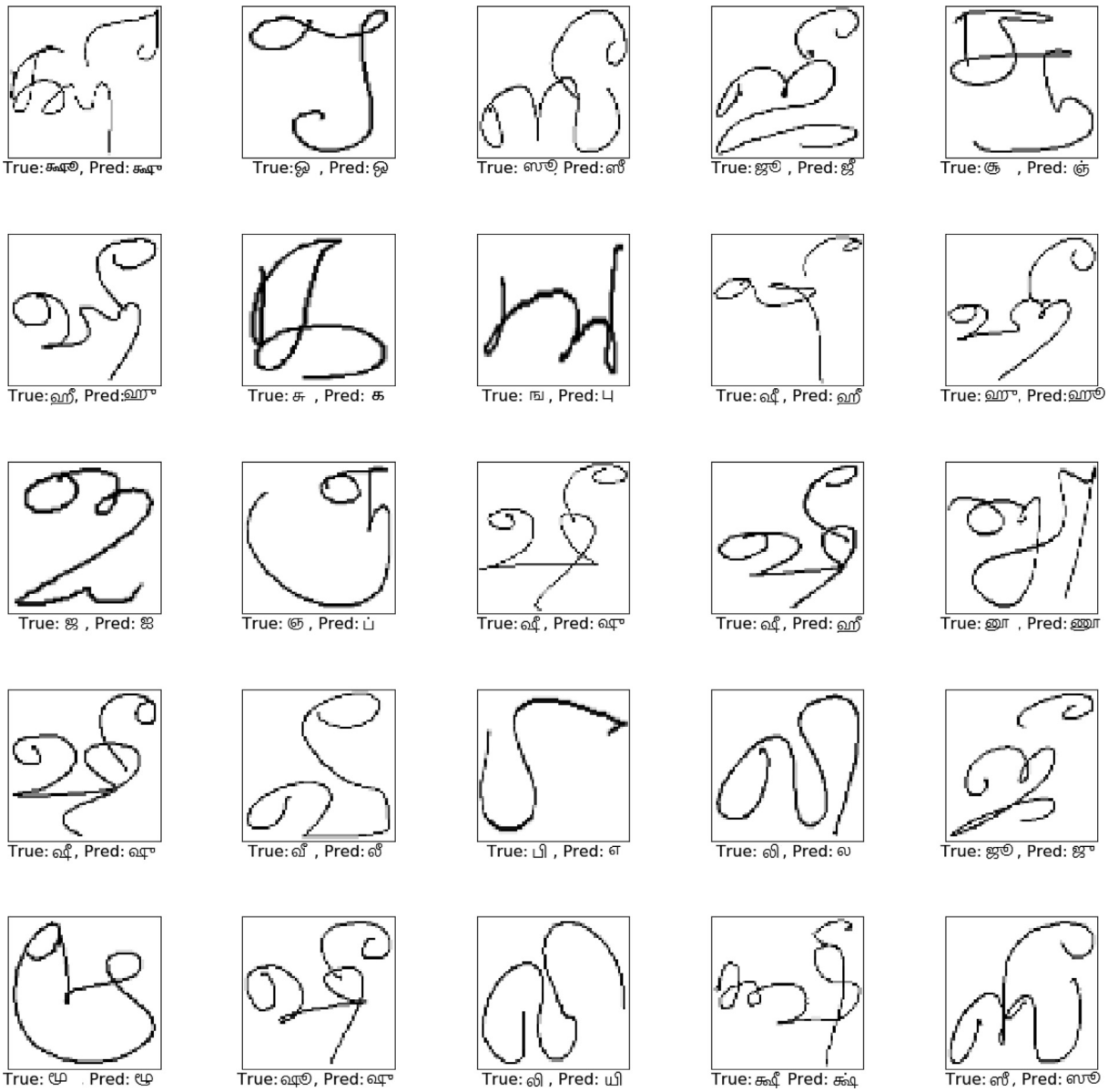

Fig. 5. Some of the wrongly recognized characters from the test set.

directly with our work. But still, their overall recognition accuracy is far less than ours. Shanti et al. used SVM classifier and trained the model for 106 classes and in testing phase only 34 classes' individual recognition of characters was presented. Vijayaragavan and Sra used CNN to classify 35 classes and have achieved 99% training accuracy and 94.4% test accuracy. However it was only for 34 classes. When compared to all these work, our work have used all the 156 classes and have achieved a training accuracy of 95.16% and a testing accuracy of 97.7%. We have also compared our work with a simple neural network constructed with seven dense layers. This network was trained with nearly 7,500,000 parameters which produced a training accuracy of 82.61% and testing accuracy of 79%.

When compared with the ground truth, we have identified the following with the wrongly recognized characters. Fig. 5 shows some of the wrongly recognized characters.

Most of them are similar characters, for example class 9 (ஓ, pronounced as *O*) and class 10 (ஓ, pronounced as *Oo*) are almost similar except for a curve at the end. Similarly class 8 (ஜ, pronounced as *I)* and class 32 (ஜ, pronounced as *Jha*) are similar. Also, the writing of the characters of class triplets (76,122,127) (ஸீ,ஸு, ஸூ), (77,123,128) (ஷீ,ஷு,ஷூ), (78,124,129) (ஜீ,ஜு,ஜூ), (79,125,130) (ஹீ,ஹு,ஹூ), (80,126,131) (க்ஷீ,க்ஷு,க்ஷூ) by the individuals were almost same as these characters were borrowed from other language and it doesn't come as part of the standard Tamil character set. Hence, these characters are often misrecognized by our system with any one of the class in the triplets.

Since we have 156 classes, we were not able to present the confusion matrix for all the classes. Hence, we have trained a smaller subset of 13 classes (12 vowels and one special character, ஃ) and the confusion matrix for the same is displayed in Fig. 6.

7. Conclusion

This paper has presented a deep learning approach for offline Handwritten Tamil Character Recognition. The results have proven that this approach have fetched good performance when compared

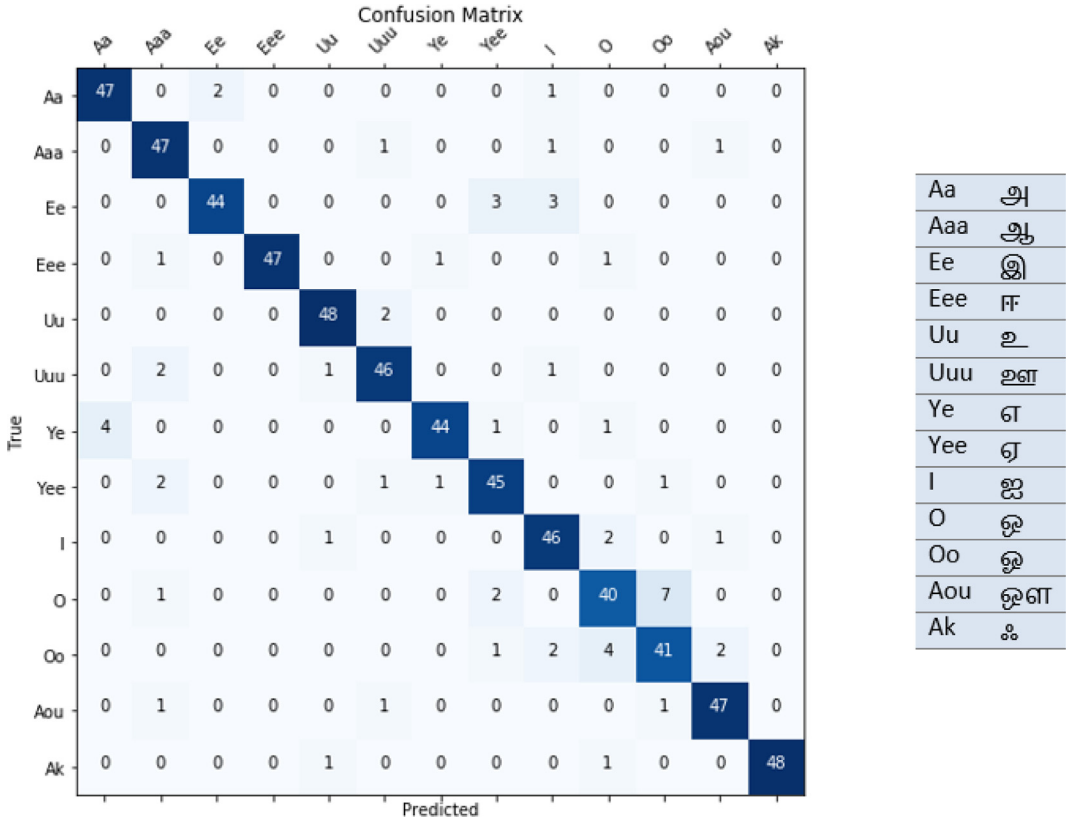

Fig. 6. Confusion matrix for a subset of 13 classes of Tamil characters, 12 vowels and one special character (ஃ).

to the traditional methods. This testing accuracy of 97.7% could even be improved by tuning the hyper parameters. Also, most of the errors were due to writing ambiguities of similar characters. The results presented here can be used as standard benchmark for HTCR research. We would like to thank HP Labs, India for having provided with a dataset that is free to use for research community.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., 2016. Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

Bhattacharya, U., Ghosh, S.K., Parui, S., 2007. A two stage recognition scheme for handwritten Tamil characters. In: 2007. ICDAR 2007. Ninth International Conference on Document Analysis and Recognition. IEEE, pp. 511–515.

Boufenar, C., Kerboua, A., Batouche, M., 2017. Investigation on deep learning for off-line handwritten Arabic character recognition. Cognit. Syst. Res.

Canziani, A., Paszke, A., Culurciello, E., 2016. An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678.

Ciregan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. In: 2012 IEEE conference on Computer vision and pattern recognition (CVPR). IEEE, pp. 3642–3649.

Cireşan, D., Meier, U., 2015. Multi-column deep neural networks for offline handwritten Chinese character classification. In: 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–6.

Elleuch, M., Tagougui, N., Kherallah, M., 2017. Optimization of DBN using regularization methods applied for recognizing Arabic handwritten script. Procedia Comput. Sci. 108, 2292–2297.

El-Sawy, A., Loey, M., Hazem, E.B., 2017. Arabic handwritten characters recognition using convolutional neural network. WSEAS Trans. Comput. Res. 5, 11–19.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249–256.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167

Jose, T.M., Wahi, A., 2013. Recognition of Tamil Handwritten Characters using Daubechies Wavelet Transforms and Feed-Forward Backpropagation Network. Int. J. Comput. Appl. 64 (8).

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105.

Liu, C.L., Yin, F., Wang, D.H., Wang, Q.F., 2013. Online and offline handwritten Chinese character recognition: benchmarking on new databases. Pattern Recognit. 46 (1), 155–162.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., 2015. Imagenet large scale visual recognition challenge. Int. J. Comput. Vision 115 (3), 211–252.

Shanthi, N., Duraiswamy, K., 2010. A novel SVM-based handwritten Tamil character recognition system. Pattern Anal. Appl. 13 (2), 173–180.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.

Sureshkumar, C., Ravichandran, T., 2010. Handwritten Tamil character recognition and conversion using neural network. Int. J. Comput. Sci. Eng. 2 (7), 2261–2267.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015, June. Going deeper with convolutions. Cvpr.

Tsai, C. Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks; Technical Report; Stanford University: Stanford, CA, USA, 2016; pp. 1–7.

Vijayaraghavan, P., Sra, M., 2014. Handwritten tamil recognition using a convolutional neural network.

Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, Cham, pp. 818–833.

Zhang, X.Y., Bengio, Y., Liu, C.L., 2017. Online and offline handwritten chinese character recognition: a comprehensive study and new benchmark. Pattern Recognit. 61, 348–360.