Telugu OCR Framework Using a HMM and Transfer Learning Approach

Jennifer N. Andriot and Venkat N. Gudivada(✉)

Department of Computer Science, East Carolina University, Greenville, NC, USA
gudivadav15@ecu.edu

Abstract. Optical character recognition (OCR) for complex scripts such as Telugu has gained much attention over the past decade due to the significant advancements made in this area of research. The Telugu OCR framework in this work proposes a hidden Markov model based approach using transfer learning to estimate the emission probability parameter of the model. This approach incorporates knowledge of the Telugu language into the framework via the hidden Markov model, while the pre-trained convolutional neural network, VGG-16, aids in estimating the emission parameters. The results from this framework show that using a pre-trained CNN for parameter estimation significantly reduces the resources and training time required for developing a Telugu OCR framework.

Keywords: Telugu · Optical character recognition · Transfer learning

1 Introduction

Telugu is a challenging script for OCR for a number of reasons. Telugu script consists of hundreds of syllables and the widths and heights of the syllables are not uniform. In addition, some characters in the script have such small differences between them that they are almost indistinguishable. These factors make developing a Telugu OCR system a difficult task with respect to all aspects of the framework, from the creation of the dataset to the segmentation process down to classification.

A novel Telugu HMM-based OCR framework is proposed in this work to address the some of the aforementioned challenges of developing a Telugu OCR framework. A combined hidden Markov model (HMM) and convolutional neural network (CNN) approach is used to incorporate information about the Telugu language into the classification process. The CNN aids to extract pertinent features from the images and to provide estimates for the emission probability parameters of the HMM.

For the classification process of the framework, the Viterbi algorithm, using MAP criterion, determines the most likely sequence of characters given the sequence of observations and outputs the most likely text in the input image.

HMMs are widely implemented for tasks such as speech recognition and biological sequence analysis [4, 9]. HMM-based OCR frameworks are also popular for character recognition of complex scripts such as Bangla, Tibetan, and Urdu [5, 6, 14]. HMM structures can be flexibly defined to adjust for different preprocessing and segmentation techniques and to incorporate information about the language that can aid in the classification process. Most recent OCR research involves the use of neural networks and deep learning for image classification.

We compare the proposed framework to an existing Telugu OCR framework, which uses a convolutional neural network trained on a dataset consisting of approximately 73,000 images [1]. Convolutional neural networks are computationally expensive to train and require sufficiently large datasets to be trained properly. To alleviate this problem, the proposed framework implements a pre-trained 16-layers convolutional neural network called VGG-16 [10], to aid in the feature extraction and parameter estimation processes. We show in this work that transfer learning alleviates the need for large image datasets and significantly reduces the training time required to develop an OCR system.

2 Related Work

Past research has shown positive results with using pre-trained CNNs for OCR. Evaluation of several pre-trained CNNs was performed for Bangla handwritten text is discussed in [2]. The models evaluated include VGG-16, DenseNet [7], ResNet [13], and FractalNet [8]. Of these four models, the DenseNet has the highest character recognition accuracy of 98.31% followed by VGG-16 with an accuracy of 97.56%.

3 Proposed Telugu OCR Framework

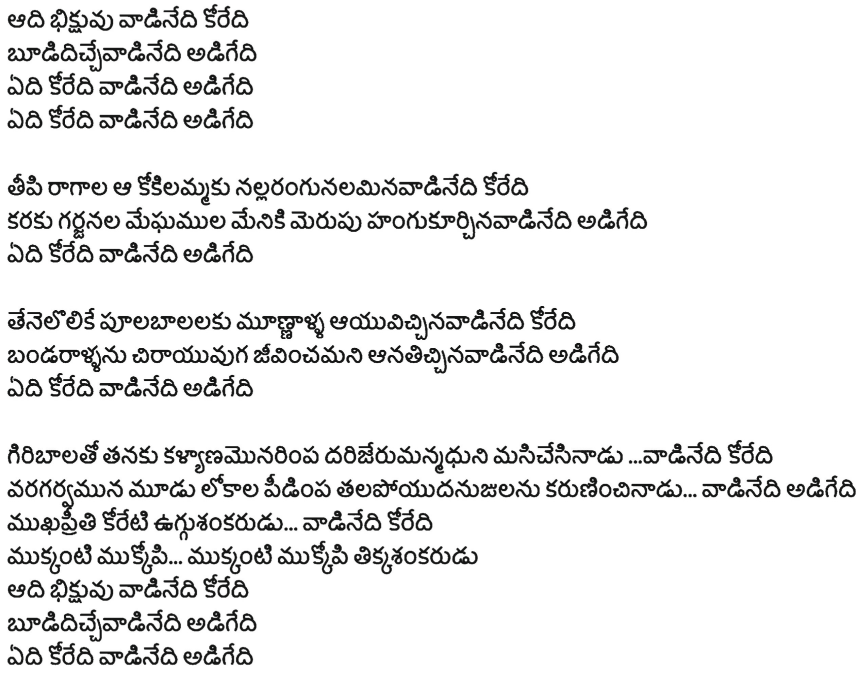
Modern Telugu script consists of 16 vowels and 36 consonants. The Telugu script is not strictly character or syllabic but a combination of both. Consonants carry an inherent vowel that can be modified by appending vowels or vowel-modifiers to the consonant. These consonant-vowel pairs produce a syllable and each syllable is written as one contiguous ligature. There are close to 500 possible combinations of consonant-vowel pairs that are commonly used in the Telugu language. A sample of Telugu script is shown in Fig. 1.

3.1 Telugu Corpus

To incorporate knowledge of the Telugu language into the OCR framework, a corpus of Telugu text was curated to estimate the transition and initial probabilities of the HMM. The corpus was created using a collection of articles from Telugu Wikipedia pages. In the preprocessing stage, non-Telugu characters and Roman characters were removed. The corpus consists of 44,856 words, 245,483 characters, and 410 Telugu syllables.

3.2 Image Dataset

The image dataset of Telugu syllables was created by extracting the set of unique syllables that exist in the text dataset and saving them to a text file. A screenshot of the text was then processed and the glyphs in the image were extracted and saved as individual images. Data augmentation techniques were applied to increase the size of the dataset and to produce slightly modified images of each extracted glyph. These methods include resizing, centering, cropping and applying horizontal and vertical shifts to the images. The dataset consists of approximately 30 images of each glyph for an approximate total of 12,000 images.


ఆది భిక్షువు వాడినేది కోరేది
బూడిదిచ్చేవాడినేది అడిగేది
ఏది కోరేది వాడినేది అడిగేది
ఏది కోరేది వాడినేది అడిగేది

తీపి రాగాల ఆ కోకిలమ్మకు నల్లరంగునలమినవాడినేది కోరేది
కరకు గర్జనల మేఘముల మేనికి మెరుపు హంగుకూర్చినవాడినేది అడిగేది
ఏది కోరేది వాడినేది అడిగేది

తేనెలొలికే పూలబాలలకు మూణ్ణాళ్ళ ఆయువిచ్చినవాడినేది కోరేది
బండరాళ్ళను చిరాయువుగ జీవించమని ఆనతిచ్చినవాడినేది అడిగేది
ఏది కోరేది వాడినేది అడిగేది

గిరిబాలతో తనకు కళ్యాణమొనరింప దరిజేరుమన్మధుని మసిచేసినాడు ...వాడినేది కోరేది
వరగర్వమున మూడు లోకాల పీడింప తలపోయుదనుజలను కరుణించినాడు... వాడినేది అడిగేది
ముఖప్రీతి కోరేటి ఉగ్గుశంకరుడు... వాడినేది కోరేది
ముక్కంటి ముక్కోపి... ముక్కంటి ముక్కోపి తిక్కశంకరుడు
ఆది భిక్షువు వాడినేది కోరేది
బూడిదిచ్చేవాడినేది అడిగేది
ఏది కోరేది వాడినేది అడిగేది

Fig. 1. Sample Telugu script

3.3 Preprocessing and Segmentation

Several preprocessing steps were applied to the image before the character recognition process. The input image was first converted to grayscale, thresholded, binarized, and inverted such that the background pixels are non-white and the foreground pixels are white. Next, a Gaussian histogram of the image was used to compute the row-ink marginals to determine the optimal row values to segment the image into lines. Word segmentation was performed in a similar manner except by using the column values to determine where to segment the words. Segmentation of the glyphs in each image of the word was performed using Tesseract's [11] bounding box function. Approximately 95% of the glyphs were correctly segmented using this process and function. Segmentation using the bounding box function alone did not produce a high segmentation accuracy. Figure 2 shows an example of the segmentation of an input image.

An additional segmentation step was applied to correct any segmentation errors. Segmentation errors that occurred were due to little separation between two glyphs in the input image. A correction was applied by computing the mean length and standard deviation of the segmented glyphs in the line. If the length of the segmented image was greater than the 68% percentile, then the image was split evenly into two.

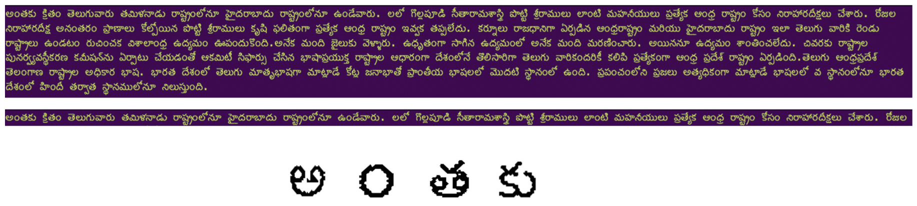


Fig. 2. Segmentation of input image: The top image shows an inverted input image. The middle image shows the first segmented line of the image. Below this are the segmented characters of the first word in the segmented line.

3.4 Training VGG-16

The architecture of VGG-16 consists of 13 convolutional layers, five max-pooling layers, two fully-connected dense layers, and an output layer using the softmax function as the activation function.

The image dataset used for fine-tuning VGG-16 was split into a training set and a validation set. Approximately 9,000 images are used for training and 3,000 images for validation. The training images were augmented in-place during training using Keras' ImageDataGenerator [3]. These augmentations included horizontal and vertical shifts to the images. Training images were shuffled before each epoch to reduce bias during training.

The VGG-16 model was then loaded using the Keras API. The first 13 layers were held frozen so as to not update the parameters of the convolution blocks during the first round of training. All 14,714,688 parameters are non-trainable parameters in this stage of training as shown in Table 1. Two dense layers were then added to the model and several node sizes for these layers were evaluated using 4096 or 1000 nodes for the first layer and 1000 or 500 nodes for the second layer. The last layer for all architectures was a softmax output layer with 410 nodes. Two optimizers, Adam and stochastic gradient descent (SGD), and two regularization methods, batch normalization and dropout, were also compared with the different architecture configurations. Early stopping based on validation loss criteria was employed to reduce over-fitting the model during training. After training the fully-connected layers on the image data set, the last convolution block of VGG-16 was fine-tuned, along with the trained dense layers, for additional epochs using the same optimizer to train the dense layers and early stopping technique. Shown in Table 2 is the architecture for fine-tuning last convolution block of the model.

The best model, shown in Table 3, is an architecture with two dense layers with 1000 and 500 nodes, respectively, using Adam optimizer and batch normalization after each dense layer.

Testing the models was performed by computing the character recognition accuracy result from an image of Telugu text. The process used to create the image dataset did not take into consideration the effect that certain characters appearing next to each other would have on the segmented result of that character. It was determined that a better evaluation of the models would be to use an actual image of text instead of individual segmented glyphs, and to choose the best model that had the highest accuracy on this text image.

3.5 Parameter Estimation

A hidden Markov model is defined by the following parameters:

- N number of states of the model
- $A = \alpha_{ij}$ transition probabilities from state i to state j
- $B = b_j(k)$ emission probabilities of an observation, O_t, being emitted from a state
- π_i initial state distribution.

The number of states, N, is defined as the number of unique syllables in the Telugu corpus, which is 410. Estimation of initial start probabilities, π, were obtained by computing the relative frequency that the syllable appeared at the start of a word in the Telugu corpus. Similarly, transition probabilities, α_{ij} were estimated by computing the relative frequency that the syllable transitioned to another syllable.

Table 1. VGG-16 trained model architecture: Two fully connected layers are added with a softmax output layer. Approximately 25.7 million parameters are trainable during the first phase of training.

Layer (type)	Output shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36 928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73 856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147 584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295 168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590 080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590 080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1 180 160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2 359 808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2 359 808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0

(*continued*)

Table 1. (*continued*)

Layer (type)	Output shape	Param #
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2 359 808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2 359 808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2 359 808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1000)	25 089 000
batch_normalization (BatchNo)	(None, 1000)	4000
dense_1 (Dense)	(None, 500)	500 500
batch_normalization_1 (Batch)	(None, 500)	2000
dense_2 (Dense)	(None, 410)	205 410
Total params: 40,515,598		
Trainable params: 25,797,910		
Non-trainable params: 14,717,688		

Table 2. VGG-16 fine-tuned architecture: The last convolution block is trained for two epochs along with the fully-connected layers from the first phase of training. The number of trainable parameters during this phase of training is 32.8 million parameters.

Layer (type)	Output shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36 928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73 856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147 584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295 168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590 080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590 080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1 180 160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2 359 808

(*continued*)

Table 2. (*continued*)

Layer (type)	Output shape	Param #
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2 359 808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2 359 808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2 359 808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2 359 808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1000)	25 089 000
batch_normalization (BatchNo)	(None, 1000)	4000
dense_1 (Dense)	(None, 500)	500 500
batch_normalization_1 (Batch)	(None, 500)	2000
dense_2 (Dense)	(None, 410)	205 410
Total params: 40,515,598		
Trainable params: 32,877,334		
Non-trainable params: 7,638,264		

To account for characters with zero starting or transitioning probability, Laplace smoothing was applied to assign a small probability to syllables with zero starting or transitioning probability. For the initial start probabilities, vowel modifiers remained zero since they do not appear at the start of a word.

The segmented glyphs were passed in as input to the VGG-16 model. A vector of 410 components was obtained for every glyph. The components in each vector were used to estimate the emission probability for the observation.

Table 3. Testing results for the top three VGG-16 architectures using the Adam optimizer and top three using the SGD optimizer. Two regularization techniques, BN and dropout, for each architecture were tested. Testing for each model was performed using an image of text and computing the character recognition accuracy.

Architecture	Optimizer	Regularization	Test accuracy
FC-1000-FC-500	Adam	BN	55%
FC-4096-FC-1000	Adam	Dropout/BN	54%
FC-4096-FC-1000	Adam	Dropout/BN	50%
FC-4096-FC-1000	SGD	Dropout/BN	52%
FC-4096-FC-1000	SGD	BN	46%
FC-1000-FC-500	SGD	BN	47%

For multi-class classification problems, the number of neurons in the last layer of a neural network is dependent on the number of classes in the dataset. The output of every neuron in the last layer can be interpreted as the likelihood that the class, c_i, was generated from the observation, o_i. Using this interpretation of the softmax function outputs, we apply Bayes' theorem to obtain emission probability estimates:

$$p(o_i|c_i) = \frac{p(c_i|o_i)p(o_i)}{p(c_i)} \tag{1}$$

Since $p(o_i)$ is the same for all observations, only the scaled likelihoods, $\frac{p(c_i|o_i)}{p(c_i)}$ are computed, where $p(c_i)$ is estimated from the Telugu text corpus.

3.6 Character Recognition

Character recognition is implemented using the Viterbi algorithm with the defined HMM, $\lambda = (\pi, A, B)$. The Viterbi algorithm determines the most likely sequence of characters from the sequence of glyphs. The recursion portion of the Viterbi algorithm is expressed as:

$$P\big(s_{1,t}, v_{1,t}\big) = \begin{cases} \pi(s_i) \cdot e(o_1|s_1), & t = 1 \\ P\big(s_{1,t-1}, v_{1,t-1}\big) \cdot \alpha(i,j) \cdot e(o_t|s_t), & t \succ 1 \end{cases} \tag{2}$$

A sequence of observations is defined as the segmented glyphs from a word. The length of a sequence is the number of segmented glyphs in the word. The function iterated over the observations using the recursive definition in Eq. 2 until it had reached the end of the sequence. At every state, at every time step, the most likely preceding state was recorded. After the last observation, the syllable with the highest probability was assumed to be the last syllable of the word. The function then backtracked through the matrix of stored states to retrieve the preceding syllables. The Viterbi algorithm was applied to each sequence of observations until all of the sequences were processed.

4 Evaluation

The following character recognition accuracy (CRA) formula is used to evaluate the proposed framework:

$$\text{CRA}\% = \frac{N - LD}{N} \times 100 \tag{3}$$

where N is the total number of characters and LD is the Levenshtein edit distance.

The proposed Telugu HMM-based OCR framework referred to as HMM-Softmax was tested and compared with an existing Telugu CNN OCR framework [1] using four test images of differing lengths and topics. We refer to existing Telugu CNN framework as CNN-Tel. Minor changes were to made CNN-Tel to resolve any deprecated function issues. CNN-Tel produces three text files when an input image is fed into the system. One file is the result without any post-processing method applied. The other two results are with post-processing, one using an n-gram language model and the other without.

The CNN-Tel model used for comparison to HMM-Softmax is the CNN-Tel model with the highest accuracy of the three that applied additional post-processing to the results without the use of the n-gram language model.

The average result over the test sets were computed for each model. Table 4 shows the results of HMM-Softmax compared to CNN-Tel. VGG-16 results are provided as a baseline for HMM-Softmax.

Table 4. Character recognition accuracy results for four input images. Each image is of varying length and topic. The average results for each model are shown in the last row of the table. The average CRA result is the result used when referencing the accuracy of a given model.

Input image	HMM-Softmax	VGG-16	CNN-Tel
1	72%	49%	71%
2	72%	55%	85%
3	73%	54%	84%
4	68%	52%	80%
Average	71%	53%	80%

5 Conclusion

The proposed Telugu OCR framework using a HMM and transfer learning approach obtained an average OCR accuracy of 71%. These results show that implementing VGG-16 as a parameter estimator for a Telugu HMM-based OCR framework may significantly reduce training time and the size of the training image dataset without significant accuracy loss.

The Telugu framework proposed in this work is specifically designed for the Telugu language. There are around twenty additional languages that use the Telugu script. One potential problem with this HMM approach is that it may not be able to be applied other languages that use the Telugu script since the transition and initial start probabilities are estimated from a corpus of Telugu text written in the Telugu language. This problem may be resolved by replacing the corpus of Telugu text used for training the HMM model with a corpus of texts in the desired language.

HMM-based OCR frameworks allow for the incorporation of the underlying distribution of the data obtained from language corpus that neural network frameworks can not easily include. Telugu script has a large number of classes which is a challenging problem for developing any type of OCR framework. It may be worth further researching methods such as pre-classification of the Telugu syllables or using a hierarchical model to first classify the image by its base character and then by the dependent character to reduce the number of states in the HMM.

Additionally, it may be worth further researching other pre-trained neural networks to determine the optimal pre-trained model to use for optical character recognition.

GoogLeNet [12] and ResNet are two additional pre-trained models that also achieve state-of-the-art performance.

In conclusion, we developed a Telugu HMM-based OCR framework using a transfer learning approach. This work has shown that transfer learning provides a less complex way to perform emission parameter estimation and reduces the resources required to train Telugu OCR models without significantly impacting character recognition results.

References

1. Achanta, R., Hasti, T.: Telugu OCR framework using deep learning. arXiv e-prints (2015)
2. Alom, M.Z., Sidike, P., Hasan, M., Taha, T.M., Asari, V.K.: Handwritten Bangla character recognition using the state-of-art deep convolutional neural networks (2018)
3. Chollet, F., et al.: Keras (2015). https://github.com/fchollet/keras
4. Dauparas, J., Wang, H., Swartz, A., Koo, P., Nitzan, M., Ovchinnikov, S.: Unified framework for modeling multivariate distributions in biological sequences (2019)
5. Hasnat, M.A., Habib, S., Khan, M.: Segmentation free Bangla OCR using hmm: training and recognition (2007)
6. Hedayati, F., Chong, J., Keutzer, K.: Recognition of Tibetan wood block prints with generalized hidden Markov and kernelized modified quadratic distance function. In: Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2034617.2034631
7. Iandola, F.N., Moskewicz, M.W., Karayev, S., Girshick, R.B., Darrell, T., Keutzer, K.: DenseNet: implementing efficient ConvNet descriptor pyramids. CoRR abs/1404.1869 (2014). http://arxiv.org/abs/1404.1869
8. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: ultra-deep neural networks without residuals (2017)
9. Pradhan, M.R.: Genome sequences analysis using HMM in biological databases. In: 2019 International Conference on Digitization (ICD), pp. 272–275 (2019). https://doi.org/10.1109/ICD47981.2019.9105756
10. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2015)
11. Smith, R.: An overview of the tesseract OCR engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633 (2007). https://doi.org/10.1109/ICDAR.2007.4376991
12. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594
13. Targ, S., Almeida, D., Lyman, K.: ResNet in ResNet: generalizing residual architectures (2016)
14. Ud Din, I., Siddiqi, I., Khalid, S., Azam, T.: Segmentation-free optical character recognition for printed Urdu text. EURASIP J. Image Video Process. **2017**(1), 62 (2017). https://doi.org/10.1186/s13640-017-0208-z