OCR for Devanagari Script using a Deep Hybrid CNN-RNN Network

Rhea Sansowa [1(✉)] [0000-0002-7928-2295], Vincent Abraham [1] [0000-0003-3336-9595], Manish I. Patel [1] [0000-0003-2948-9770], and Ruchi Gajjar [1] [0000-0002-8136-6084]

[1] Department of Electronics and Communication Engineering
Institute of Technology, Nirma University, Ahmedabad, India
rhea.s.sansowa@gmail.com, vincent.ahm@gmail.com,
manish.i.patel@nirmauni.ac.in, ruchi.gajjar@nirmauni.ac.in

Abstract. OCR (Optical Character Recognition) involves the electronic transcription of handwritten, printed text from either scanned documents or any sort of images. This multiclass classification problem comprising of recognizing the various characters in a language and correctly classifying them has found its way in plenty of computer vision applications. Although much work has been done for the English language, there have been only a few explorations pertaining to OCR for the Devanagari script. This paper proposes a hybrid CNN-RNN model to classify characters using the Devanagari Handwritten Character Dataset. The main objective is to design a model with higher accuracy than the CNN model reported in literature for the same purpose. The models are trained and evaluated using the same procedures. On evaluating the models, the hybrid CNN-RNN model has a testing accuracy of 98.71%, which is higher than the CNN model, having 97.71% testing accuracy. It also fares better than the standard neural network architectures – VGG16 and AlexNet which when taken without the pre-trained weights result in 97.62% and 98.20% testing accuracy respectively. Hence, this successfully demonstrates the attributes of RNN in improved feature extraction when used along with convolutional layers.

Keywords: Convolutional neural network, Devanagari script, Hybrid CNN-RNN, Long short-term memory, Optical character recognition, Recurrent neural network.

1 Introduction

Optical character recognition (OCR) is a technology that can be used to convert any scanned image with natural language into a searchable document, which can be edited and modified, reducing the searching time from hours to mere minutes. It also has applications in other fields such as licence plate reading, digitizing historical documents and books. The scanned text can also be used for text mining.

The preliminary idea of OCR surfaced well before the age of computers around 1950. Tausheek, in the year 1929, obtained a patent on the technology of template

masking and matching, which was made using photodetectors and mechanical masks [1]. The research work done by Anderson R.H. in 1967 paved the way for more work in pattern extraction from images and its conversion into markup form [2]. With the increase in computational power of systems, the availability of GPU, and advents in deep learning, there has been a boost in OCR technology, and new neural network-models are now available for it. Deep learning models are ideal for OCR because feature selection is a key task [3] and in deep learning models, the features are built by the model itself, thus reducing efforts and improving classification [4]. Convolutional neural network (CNN) is one such neural network class. It is easier and less time consuming to train as number of parameters are relatively low compared to classes having fully connected layers [5]. CNN is most commonly used in such computer vision applications, while much use of RNN is not seen. Performance of this CNN model can be boosted by passing the extracted features by CNN into the RNN model for further classification [6]. CNN is used for feature extraction because it has robust feature extraction ability [7]. Substantial work has been done for OCR in English language, but there's still enormous scope for research related to Devanagari script. The main reason behind this is that there is no separation between characters in the script, and a lot of the characters look alike with only a slight distinction between them [8]. This paper focuses on improving the CNN model found in the existing literature [8] by adding RNN layers for further feature extractions and implementing to carry out OCR for Devanagari script with increased accuracy. These values are then compared with the existing architectures – AlexNet and VGG16 to gauge the performance of the proposed model.

1.1 The Devanagari Handwritten Character Dataset

Devanagari belongs to the Brahmic script family, and it makes up languages such as Hindi, Nepali and Marathi, among other East and South Asian languages [11]. There are 36 consonants in this script and a total of 12 vowels and 10 numbers. The vowels can also attach themselves to the consonants, resulting in derived consonants. There are some other special characters too found in this script.

The dataset chosen for this paper is called 'Devanagari Handwritten Character Dataset' (DHCD), and it has all the 36 consonants and the 10 numerical digits as a part of the dataset [8]. It does not contain vowels, modified characters or special characters. In Table 1 and Table 2, the characters and numerical digits in the Devanagari script contained by the dataset are displayed.

Table 1. Devanagari consonants

क	ख	ग	घ	ङ	च	छ	ज	झ	ञ
ट	ठ	ड	ढ	ण	त	थ	द	ध	न
प	फ	ब	भ	म	य	र	ल	व	श
		ष	स	ह	क्ष	त्र	ज्ञ		

Table 2. Devanagari numerical digits

०	१	२	३	५	६	७	८	९	४

There are 92,000 images - 78,200 for training and 13,800 for testing. The size of each PNG image is 32 x 32. The dataset is made of handwritten characters that were manually cropped, and characters were placed in the center of the image and were of size 28x28. 2x2 padding of zeros was added on all sides of the characters. Sample images from the dataset are shown in Fig. 1.

Fig. 1. Sample images from the Devanagari Handwritten Character Dataset.

Section 2 contains details about some of the research work done till date relevant to this topic, section 3 contains basic theory about CNN and RNN after which in section 4, the hybrid CNN-RNN model is proposed by making modifications to the CNN model reported in literature. The implementation details are covered at the beginning of section 5 and results for all the models have been stated and interpreted. The paper is concluded in section 6.

2 Literature Review

A fair amount of work has been done on image classification for various applications in fields such as OCR, healthcare, obstacle recognition, etc. using methodologies which are novel, or are optimizations on existing ones. Some of these literary works relevant to this paper's purpose would now be discussed in this section.

A dataset was introduced by [8] for the purpose of OCR for Devanagari script. Two CNN models were also proposed in the paper. The first model consisted of three convolutional layers and a fully connected layer. Based on the LeNet family, the second model consisted of two convolutional layers followed by two fully connected layers. To optimize the training process, methodologies like dropouts, batch normalizations and dataset increment were applied to the models, which led to the models achieving a testing accuracy of 98.47% and 98.26%. Hence, a dataset for the models to be trained on and a base CNN model was provided [8]. One of the more recent papers to use this dataset used a deep CNN model through which an accuracy of 97% was observed [21]. A hybrid CNN-RNN model for implementing Urdu OCR from printed documents was presented, annotated page by page [9]. It involved sequence to sequence transcription of the Urdu text, which was later divided at line levels and prepared into a dataset. The proposed model was a 7-layer deep convolutional network. The convolutional layers

were used to extract the robust features from the input which were later mapped and fed into two BLSTM layers for the final transcription. Different variations in this model were also proposed in which the highest testing accuracy was achieved as 98.80%.

To carry out the task of image to markup generation, WYGIWYS model was built which used an attention-based mechanism instead of taking the standard mathematical approach to OCR [20]. After feature extraction was done using CNN, the grid was fed into an RNN encoder after which on decoding, using the attention-based mechanism, result is mapped to the vocabulary. The 'Image to Latex 100K' dataset was used for this purpose. Its implementation yielded 74.00% accuracy. This was improved upon by building a complex attention model with CNN-LSTM for the particular purpose of recognition of various mathematical equations [10]. The proposed model had worked sequentially to read data which aided in keeping track of the data processed, which ultimately resulted in better performance with a higher accuracy of 76.00%. A parallel CNN-RNN model was proposed for the classification of breast biopsy images non-invasively into four classes to find type of cancer cells present in which both CNN and RNN were used for feature extraction, and an assumption was made about RNN having the same feature extraction ability as CNN in [7]. For weighing the features, a perceptron attention mechanism was used so that the model can be improved. Instead of conventional dropout, selective dropout, which uses pruning to discard the neurons, was used. The highest test accuracy was observed to be 97.50%.

The papers reviewed in this section show a clear benefit of adding RNN layers to a CNN model for image classification applications, resulting in a model with more accuracy. Furthermore, the models described in [7] and [9] acted as strong motivation and justifications in using RNN along with CNN for further feature extractions from input.

3 Background Theory

This section would talk about the various methodologies used for the purpose of this paper, and also the implementation details for achieving the desired objective.

3.1 Convolutional Neural Networks (CNN)

A convolutional neural network plays a vital role in robust feature extractions of the fed in input and hence is a convenient choice for OCR. Its foundation was laid when Fukushima et al., 1980 came up with the Neocognitron model for visual pattern recognition [17]. Through its training process, which usually consists of backpropagation through gradient descent, a CNN can extract local features of the input adaptively, without any need for explicitly stating the features to be learnt. A CNN generally consists of convolutional layers, max-pooling (sub-sampling) layers & fully connected (dense) layers. The convolutional and max-pooling layers are responsible for feature extraction. In contrast, the dense layer maps the extracted features to a given number of nodes for the terminal classification.

3.2 Recurrent Neural Networks (RNN) & Long Short-Term Memory (LSTM)

In recurrent neural networks, the current output depends upon the current and previous inputs fed into the network. This attribute of past dependency has allowed RNNs to be used for various applications involving time-series predictions and exploring the temporal dynamics between the information [18]. RNNs achieve this dependency by having the output at each timestep be the weighted sum of the current inputs, added with the activation output of the previous node. The training process here takes place through backpropagation through time. RNNs face the issue of not maintaining the long-term dependencies between the data due to vanishing or exploding gradients. To overcome this, LSTM is used wherein a gated cell is present that stores or forgets the previous input information based on its weights. Hence, this additional gradient of the gated cell allows the network to have long-term dependencies between the inputs. In this paper, LSTM layers would be used along with CNN layers to perform further feature extractions and train a model with higher accuracy.

4 The Proposed Hybrid CNN-RNN Model Architecture

This section discusses the architecture of the proposed CNN-RNN model which is designed by extending the CNN model referred from literature [8]. The architecture of the proposed model is shown in Fig. 2.

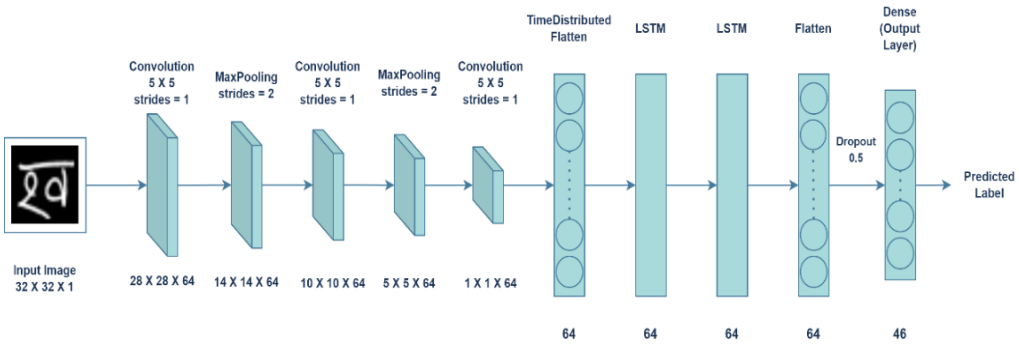

Fig. 2. The proposed hybrid CNN-RNN model

The convolutional layers in the model consist of kernels of size (5x5), and the max-pooling layers consist of kernels of size (2x2) and strides of 2. The features extracted from the three convolutional layers are flattened, time-mapped and later fed into the LSTM layers, which perform the final feature extraction before feeding them into the fully connected layers.

The activation function used for the convolutional layers is the ReLu (Rectified Linear Unit) function which responds non-linearly to negative inputs and linearly to the positive inputs. Hence, it overcomes the issues faced by other activation functions such as vanishing gradients wherein, the gradients move closer and closer to zero the further

they move from the output nodes during backpropagation. Each of the LSTM layers use the sigmoid activation function for the input, output and forget gates, allowing the gates to store or lose the information (0 or 1). The output nodes in the layers use the hyperbolic tangent (tanh) activation function, which provides the range (-1, 1), acting as an optimum activation function for backpropagation. The activation function used for both the dense layers is the softmax function, which takes the input vector and maps it to its corresponding probabilities, which for the output layer provides the information on what character the input is more likely to be. The character with the highest probability is then taken as the predicted output.

Furthermore, some essential methodologies used here for efficient training and to avoid overfitting in the model are batch normalizations and dropouts. Batch normalization refers to standardizing the inputs corresponding to each mini-batch for the subsequent layers, allowing the networks to train faster. Dropouts refer to dropping out some randomly selected neurons (based on the dropout rate) by assigning their weight values zero. Dropouts reduce overfitting and allow the trained models to be more robust [14].

By increasing the number of filters in the convolutional layers, change in the dropout rate, using the softmax activation function for both of the dense layers, and addition of the LSTM layers, performance of the referred CNN model was improved.

5 Results And Discussion

This section discusses the procedure followed for implementation of the models as well as the results observed on deploying the dataset on the defined models.

5.1 Implementation Details

Firstly, the data is loaded into the Python environment, where basic pre-processing operations are performed on the images. The images are turned into gray-scale to reduce the number of channels so that the training process can be sped up, and reshaping is done on the image where its size goes from (32, 32) to (32, 32, 1) to show the number of channels. Then assignment of labels is carried out which are then converted into one hot encoded form.

The next step involves defining the model architectures. The CNN-RNN model is defined following the architecture described in the previous section, along with the rest of the models. For validation, 30% of the training set is used wherein at the end of each epoch, the validation accuracy is monitored and the necessary decisions are made using callbacks. The callbacks used include stopping the training process on detecting no significant change in validation loss as well as reducing the learning rate on detecting a plateau. After completion of the training process, training and testing accuracy is obtained. The accuracy vs. epochs graph and metrics such as precision, recall and F1 score

are then obtained for the models. The models are then tested against a set of custom input images. For its pre-processing, images are resized, inverted & rescaled to be given as the input. The predicted output for these images is then observed.

5.2 Accuracy vs. Epochs

The models were trained by applying early stopping and using a function to reduce learning rate when the model stops improving. This gave the models ample amount of time for fine-tuning, resulting in high accuracies.

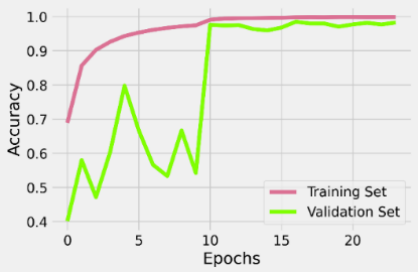

Fig. 3. Accuracy vs. Epochs for AlexNet model

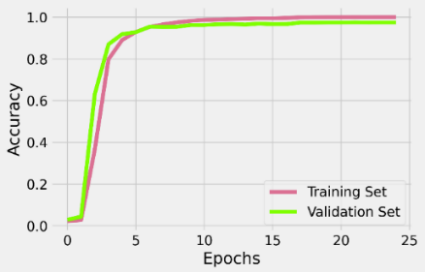

Fig. 4. Accuracy vs. Epochs for VGG16 model

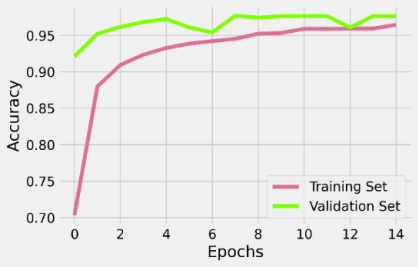

Fig. 5. Accuracy vs. Epochs for CNN [8] model

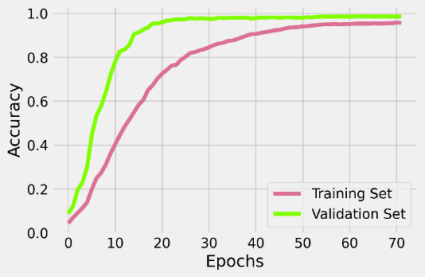

Fig. 6. Accuracy vs. Epochs for the Hybrid CNN-RNN model

The plotted results for AlexNet, VGG16, CNN [8] and proposed CNN-RNN model are as shown in Fig. 3, Fig. 4, Fig. 5 and Fig. 6 respectively. From these figures, it can be observed that all of the models achieve considerably high values of training and testing accuracies, the CNN-RNN model achieving the highest.

5.3 Quantitative Results

Accuracy of the proposed CNN-RNN model has been compared with the existing CNN model in addition to two of the standard CNN models – AlexNet and VGG16. Alexnet was one of the first deep neural networks to achieve classification accurcy comparable to traditional methods for ImageNet dataset using CNN [12]. VGG16 on the other hand

boosts the performance of AlexNet by converting large filters into multiple smaller ones, which increases the depth of the network and ultimately, improves the performance [13]. Training and testing for these two models is carried out without using the pre-trained weights and just the model architecture is utilized.

Table 3. Accuracy obtained for the models

Model Name	Accuracy		Trainable Parameters
	Training	Testing	
AlexNet	99.48%	98.20%	23,628,846
VGG16	99.25%	97.62%	14,738,286
CNN [8]	98.93%	97.71%	85,262
CNN [21]	97%		Not mentioned
Hybrid CNN-RNN	99.63%	98.71%	280,174

From the obtained values in Table 3, it can be said that the proposed CNN-RNN model outperforms the CNN, VGG16 and AlexNet models in terms of both training and testing accuracy. It is also to be noted that the CNN-RNN model has about 50 times lesser training parameters in relation to the standard models taken yet it outmatches their performance. The CNN-RNN model also outperforms the CNN model proposed in [21] which had an accuracy of 97%.

For the evaluation of models, the standard metrics - precision, recall and F1 score are used. Precision is the fraction of the true positive cases predicted from all positive cases. The fraction of real positive cases which were correctly forecast as such is called the recall value [15]. The F1 score is the harmonic mean of precision and recall [16]. As this is a multi-class problem, we use macro values for all the above-mentioned measures which results into the average value of the metric per class [15]. The macro precision, recall and F1 score values obtained are shown in Table 4.

Table 4. Macro Precision, Recall and F1 score for the models

Model Name	Precision	Recall	F1 score
AlexNet	0.98245	0.98245	0.98225
VGG16	0.97548	0.97536	0.97537
CNN [8]	0.97731	0.97703	0.97706
Hybrid CNN-RNN	0.98712	0.98703	0.98704

5.4 Dealing With Custom Inputs

Output was obtained for some images which were not present in the training and the testing dataset to see if the constructed models work on real-life data. Fig. 7 shows the steps involved to deploy the model on custom images.

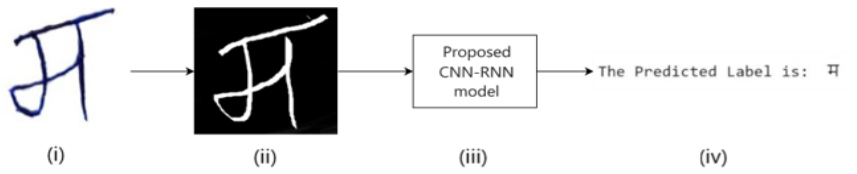

Fig. 7. Classification result of custom input (i) Image of scanned character, (ii) Pre-processed image, (iii) Passing image through the proposed CNN-RNN model, (iv) Predicted output label

6 Conclusion

The accuracy for OCR for Devanagari script can be enhanced by adding recurrent neural network layers to the CNN model. By making a hybrid model, the classification capability increases, and the misclassification rate drops down. The precision and recall values obtained are also higher than the CNN model reported in literature. The CNN-RNN model performed better than CNN, VGG16 and AlexNet, with the training and testing accuracy of 99.63% and 98.71% respectively. The number of trainable parameters for this model was about 50 times lesser than AlexNet and VGG16 and yet superior performance was obtained. For future work, a more inclusive data set containing vowels and special characters can be adopted for a more practically applicable transcription.

References

1. Mori S, Suen C, Yamamoto K (1992) Historical review of OCR research and development. Proceedings of the IEEE 80:1029-1058. doi: 10.1109/5.156468.
2. Bahdanau, D, Cho K, Bengio Y (2015). Neural Machine Translation by Jointly Learning to Align and Translate. ICLR arXiv:1409.0473.
3. Ruck D, Rogers S, Kabrisky M, Mills J (1990) Multisensor fusion classification with a multilayer perceptron. 1990 IJCNN International Joint Conference on Neural Networks 2:863–868. doi: 10.1109/ijcnn.1990.137802
4. Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09 609–616. doi: 10.1145/1553374.1553453
5. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86:2278–2324. doi: 10.1109/5.726791
6. Guo Y, Liu Y, Bakker EM, et al (2017) CNN-RNN: A large-scale Hierarchical image classification framework. Multimedia Tools and Applications 77:10251–10271. doi: 10.1007/s11042-017-5443-x

7. Yao H, Zhang X, Zhou X, Liu S (2019) Parallel structure deep neural network using CNN and RNN with an attention mechanism for breast cancer histology image classification. Cancers 11:1901. doi: 10.3390/cancers11121901.
8. Acharya S, Pant AK, Gyawali PK (2015) Deep learning based large scale handwritten devanagari character recognition. 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA) 1–6. doi: 10.1109/skima.2015.7400041
9. Jain M, Mathew M, Jawahar C (2017) Unconstrained OCR for Urdu Using Deep CNN-RNN Hybrid Networks. 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR). doi: 10.1109/acpr.2017.5.
10. Mishra V, Kaur D (2018) Sequence-to-Sequence Learning Using Deep Learning for Optical Character Recognition (OCR). 2018 International Conference on Computational Science and Computational Intelligence (CSCI). doi: 10.1109/csci46756.2018.00069.
11. Fischer SR (2004) In: History of writing. Reaktion Books, London, pp 32–37
12. Krizhevsky A, Sutskever I, Hinton G (2017) Imagenet classification with deep convolutional neural networks. Communications of the ACM 60:84-90. doi: 10.1145/3065386.
13. Simonyan K, Zisserman A (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR arXiv:1409.1556.
14. Park S, Kwak N (2017) Analysis on the dropout effect in convolutional neural networks. Computer Vision – ACCV 2016 Springer, Cham, 189–204. doi: 10.1007/978-3-319-54184-6_12
15. Paredes-Valverde M, Colomo-Palacios R, Salas-Zárate M, Valencia-García R (2017) Sentiment analysis in spanish for improvement of products and services: a deep learning approach. Scientific Programming 2017:1-6. doi: 10.1155/2017/1329281.
16. Salas-Zárate Mdel, Valencia-García R, Ruiz-Martínez A, Colomo-Palacios R (2016) Feature-based opinion mining in financial news: An ontology-driven approach. Journal of Information Science 43:458–479. doi: 10.1177/0165551516645528
17. Fukushima K (1980) Neocognitron: A self-organising neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36:193–202. doi: 10.1007/bf00344251
18. Sherstinsky A (2020) Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Elsevier journal “Physica D: Nonlinear Phenomena”, Volume 404, March 2020: Special Issue on Machine Learning and Dynamical Systems, 404:132306. doi: 10.1016/j.physd.2019.132306.
19. Prechelt L (1998) Automatic early stopping using cross validation: Quantifying the criteria. Neural Networks 11:761–767. doi: 10.1016/s0893-6080(98)00010-0
20. Deng Y, Kanervisto A, Rush A (2016) What You Get Is What You See: A Visual Markup Decompiler. Association for the Advancement of Artificial Intelligence:32-37. ArXiv, abs/1609.04938
21. Kavya A, Vivek N, Harika M, Nidumolu V (2020) Handwritten Devanagari character classification using CNN. Lecture Notes in Networks and Systems 145:309–317. doi: 10.1007/978-981-15-7345-3_25.