Hindi Handwritten Character Recognition using Deep Convolution Neural Network

Deepak Chaudhary
Deptt. of Electronics and Communication Engineering
Ambedkar Institute of Advanced Communication Technologies and Research
New Delhi, India
deepakvats97@gmail.com

Kaushal Sharma
Deptt. of Electronics and Communication Engineering
Ambedkar Institute of Advanced Communication Technologies and Research
New Delhi, India
kaushalsharma880@gmail.com

***Abstract*— Convolution Neural Network (CNN) is turning out to be a very powerful tool for solving Machine Learning (ML) problems, especially in multiclass image classification. With the availability of a huge handwritten dataset, it is possible to achieve a never thought machine accuracy in image classification. In this paper, we have proposed a Deep Convolution Neural Network (DCNN) for Hindi handwritten character recognition. The idea is basically an expansion of LeNet-5 architecture. We have trained our model using 96000 character sets, obtaining a validation set accuracy of 95.72 per cent using Adam optimizer and 93.68 per cent using RMSprop optimizer.**

Keywords—deep convolutional neural network; CNN; hindi handwritten; character recognition; max-pool; softmax classifier; machine learning

I. INTRODUCTION

Handwritten recognition is turning out to be a very interesting research field because of the huge access to the data. Hindi is one language where the scope of research is still widespread, with it being one of the most popular languages in the world. Character recognition of Hindi character is a little bit difficult as compared to character recognition of other languages like English, because of similarities in the characters, e.g. अ, अः. We propose DCNN-based model for Hindi handwritten character recognition, which shows significant accuracy.

DCNN [1] is has been proved to be one of the techniques for solving character recognition problems [2]. Different classical architectures are available on CNN. Some of the commonly used architectures are ImageNet [3], Inception Network [4], LeNet-5[5], etc.

The proposed model uses a well compiled available dataset [2] and is tested with a dataset containing 96,000 characters in the classical LeNet-5 with 46 output class which gave poor results and was only attaining the validation test accuracy of 79.35%. With LeNet-5, one more convolution layer was added and then a max pooling layer in between to build the proposed CNN model. The model gave a validation set accuracy as high as 95.72% and 93.68% using Adam and RMSprop optimizer respectively.

II. RELATED WORKS

In recent literature, there are major researches conducted in handwritten classification problem.

Verma [6] provided a model of a neural network using multilayer perceptron and radial basis function. The results of MLP and RBF were also compared by the author. In [7], Hanmandlu et al. proposed a fuzzy model-based recognition of Hindi characters. The authors used some kind of reinforcement learning and thereby enhancing the training improvement by 25-fold. Authors observed the recognition rate 90.65%.

Singh and Lehri [8] designed an offline handwritten recognition using neural network]. The authors emphasized on the preprocessing part and then it was fed to the neural network, thereby achieving an accuracy of 93%.

Singh et al. [9] implemented a model using neural network with gradient feature extraction technique, which provided more than 94% recognition accuracy. Authors also performed a comparative analysis of both global input and gradient feature input.

Jawahar et al. [10] proposed a model using principal component analysis (APC) followed by the support vector classification. This PCA analysis reduced the dimensions and speeds up the algorithm. This model achieved an overall accuracy of 96.7%.

Acharya [11] proposed deep learning based large scale handwritten Devanagari character recognition. Jangid and Srivastava [12] developed a handwritten Devanagari character recognition system using layer-wise training of DCNN and adaptive gradient methods. ISIDCHAR database was used to evaluate the system. The results of layer-wise-trained DCNN were observed favorable.

For recognizing handwritten Arabic numeral, Ashiquzzaman and Tushar [13] proposed a model using using deep learning neural networks. Using multiple classifiers, Yadav and Purwar [14] proposed an approach for handwritten characters.

The literature reveals that a number of neural networks models have been proposed for character recognition [15] [16] [17] which helped us to propose our model.

III. TERMINOLOGIES

A. Convolution Neural Network

CNN and Neural Networks (NN) both have learnable parameters and bias terms. In a convolutional network, we basically have three types of layers: convolution layer,

978-93-80544-34-2$31.00 ©2019 IEEE

pooling layer and fully connected layer. Neural Networks have fully connected layers and if we have to apply Neural Networks in image classification problem then it will not be feasible to learn so many parameters and hence resulting in slow computation and sometimes overfitting too. Hence, we have used CNN which have the advantage of parameter sharing and sparsity of connections.

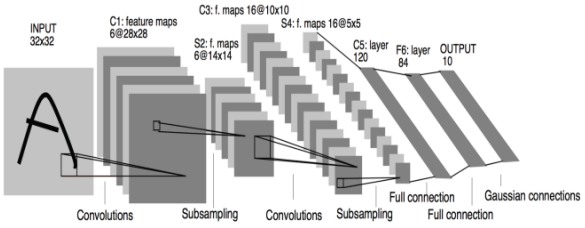

Fig. 1. A Convolution Neural Network [18]

Figure 1 demonstrates an example of a deep convolutional neural network presented in the original paper of LeNet-5 in 1998 [18]. The model has 60,000 parameters to learn, conv-pool-conv-pool-FC-FC-output network flow, and used Sigmoid/tanh activation function.

B. Convolution Layer

Figure 2 demonstrates an example of convolution operation from a 3*3 filter and the output of the scalar product is projected in the output volume.

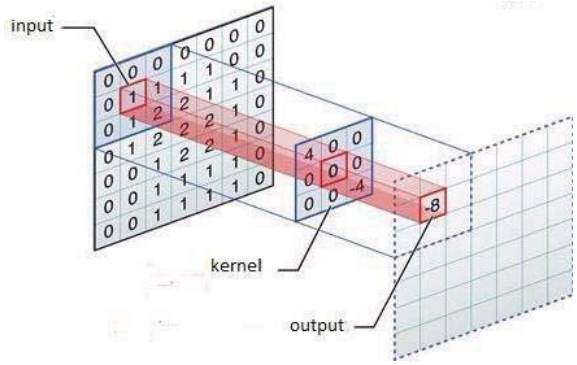

Fig. 2. Convolutional Operation Visualization [19]

Convolution layer is the heart of CNN which reduces the trainable weight parameters and also allows us the parameter sharing in one feature map. The filter slides over the input volume. In one slide it takes the dot product of all the overlapping pixels and projects this scalar result into output volume. The number of channels of the input image and the filter must be the same.

There are mainly two reasons to use this layer. First one is parameter sharing and the second one is sparsity of connections. If we simply apply fully connected layer then it would we infeasible to train such a big neural network.

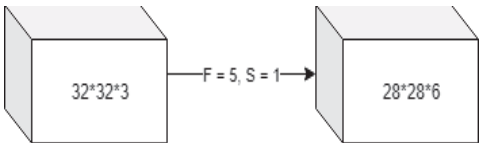

Fig. 3. Input and output volumes after convolution

If we apply fully connected layer in the above volumes, the no of parameters to learn will be 32*32*3*28*28*6, which is nearly equal to training 14 million features, which is too big and even bigger when we have a deeper neural network in which we have several layers. But in the same model if we use the CNN then the no of parameters to learn will be (5*5+1)*6 = 156, which seems practical to train.

In convolution, we have parameters sharing feature which is very useful, if we have an edge detector in previous layers then this can also be used in the later layers, so when the single filter is convolved its feature is shared in the whole volume.

If an image of h * w * c is convolved with f * f * c filter, the resulting output volume will be of the size as expressed in Equation (1).

$$\left\lfloor \frac{h-f+2*p}{s} + 1 \right\rfloor * \left\lfloor \frac{w-f+2*p}{s} + 1 \right\rfloor * n_f \qquad (1)$$

where,

n_f = number of filters used,

s = stride, and

p = padding.

C. Pooling Layer

To reduce the size of the input feature matrix, we usually use pooling layer. It speeds up the computation. Many experiments have shown that it really enhances the performance of the system. There are no parameters to learn in pooling layer. There are a lot of pooling techniques available but some commonly used are Maxpool and Average pool. We often use 2x2 submatrix for pooling which eliminates 75% data in one sampling process. We will be using Maxpool in this model.

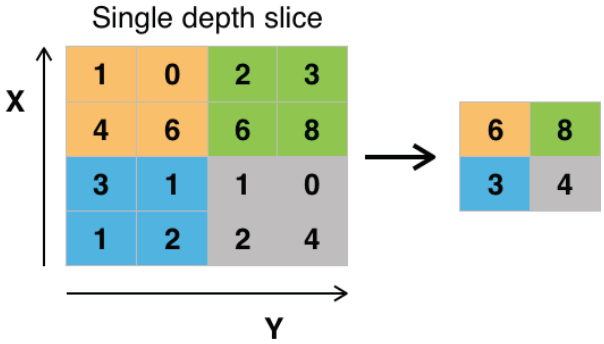

Fig. 4. Before and After Max-Pooling

Figure 4 shows the max pooling in the input image and projects the maximum of all 2*2 coloured submatrix and projects it into the output

If the input image is w * h * n_c and pool size is f * f then output volume will be of the size as expressed by Equation (2).

$$\left\lfloor \frac{w-f}{s} + 1 \right\rfloor * \left\lfloor \frac{h-f}{s} + 1 \right\rfloor * n_c \qquad (2)$$

where

s = stride.

D. Softmax Classifier

Softmax classifier is a more generalized form of the binary classifier in logistic regression. It is also known as normalized exponential function. Since our main objective is to classify among different output classes, Softmax classifier, in this case, takes input a linear vector (46 in our case) and normalizes it into probability distribution containing 46 probabilities. Hence every component is in range (0, 1). Softmax is most commonly used in neural networks, to map the un-normalized output of the penultimate layer of the network to probability over predicted output classes.

The Softmax function $f : R^k \rightarrow R^k$ is defined in Equation (3).

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (3)$$

$$for\ j = 1, .., K \text{ and } z = (z_1, \dots, z_k) \in R^k$$

IV. Proposed Work

A. Model Overview and Architecture

The motivation for the proposed model is taken from classical LeNet-5 CNN architecture. By current standards, this LeNet-5 architecture is very simple. With the availability of huge computational power today, we can have a deeper layer without fearing about the performance of the model. We have embedded an extra same convolution and pooling layer in between and used Max pooling instead of the proposed average pooling in original LeNet-5 architecture. With the addition of this extra convolution and pool layer, the performance of the system increased to quite an extent.

We have, in total, 9 layers and 46 output classes in Softmax classifier. The vowels are missing in the dataset hence 10 numeral + 36 consonants = 46 characters are trained using this model. The proposed architecture of the model is described in Figure 5 where every arrow denotes the output size after performing the operation described inside the box.

B. Model Description

- In every layer of Convolution, we have used ReLU [20] non-linearity activation function (as shown in Figure 6). ReLU is Rectified Linear Unit defined using Equation (4)

  $$f(x) = \max(0, x) \quad (4)$$

  This helps us learn some complicated functions in Deep Neural Networks. ReLU reduces the likelihood of vanishing gradient and increases sparse representation.

- Cross-entropy [21] is used as a loss function which is defined in Equation (5).

  $$h_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(p(y_i) + (1 - y_i) * \log(1 - p(y_i))) \quad (5)$$

  Softmax layer provides us with the probability distribution of different classes and to measure the performance of this classification model whose output is the range of (0, 1) we generally use this cross-entropy loss or log loss. Cross-entropy increases as the predicted probability diverges from the actual level. A perfect model will ideally have log loss of 0.

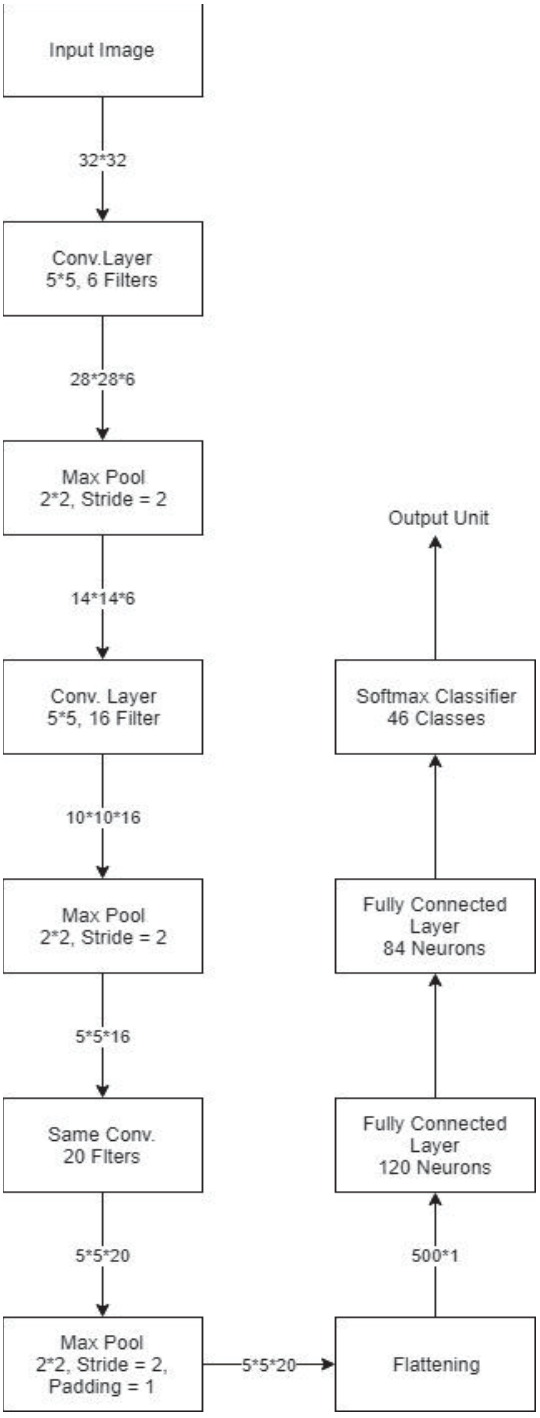

Fig. 5. Model Architecture

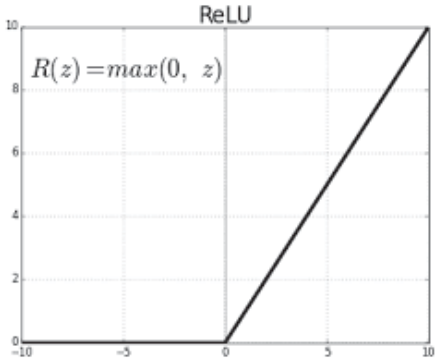

Fig. 6. ReLU Diagram

- We have used two optimization techniques, Adam and RMSprop optimizer. The objective of all the optimizers is to reach the global minima thereby reducing the cost function to its minimal value. Earlier the models were using techniques like gradient descent, but with the advancement of the different techniques, this method is rarely used nowadays. Gradient descent with momentum seems to converge faster than the original gradient descent. We will be using RMSprop optimizer which is very much similar to the gradient descent with momentum with a slight modification. As a second model, we will also be using Adam optimizer which is the mixture of RMSprop and gradient descent. Adam optimizer found to just perform better than RMSprop by 2.04%. These two optimization techniques work better than classical gradient descent optimization.

- 2 convolutional layers of a 5x5 kernel with 6 and 16 filters are used. Then we used same-convolution and pooling layer to retain the input height and width. We will obtain 5x5x20 matrix after these convolution and pooling operations. We will flatten this 5x5x20 into one vector of size 500x1, which is then connected to the fully connected layer having 120 neurons, which is then followed by another fully connected layer having 84 neurons.

  This model gives 46 scalar outputs using a softmax classifier. We predict the output as the classifier which has maximum value among all of them.

  More formally, we output the class j with the objective, as given in Equation (6)

$$\max(p(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{46} e^{z_k}} \ \forall\ j \in (1,46)) \qquad (6)$$

V. Results and Discussions

The model is trained using 96,000 character sets using two optimizers Adam and RMSprop which respectively gives 95.72% and 93.68% accuracy in the validation set. Thus Adam optimizer is found to be working slightly better than RMSprop. We split the data into 70:20:10 ratio. 70% for the training set, 20% for cross-validation and 10% for the test set. We carried out the training using 25 epochs and noted the accuracy after each epoch. The accuracy seems to increase drastically in the initial epochs and gets saturated in the higher iterations. On the other hand, model loss was always declining as shown in Figure 7 and Figure 8.

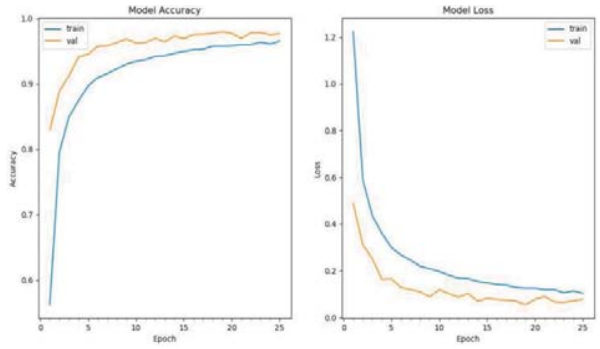

Fig. 7. Results with Adam Optimizer

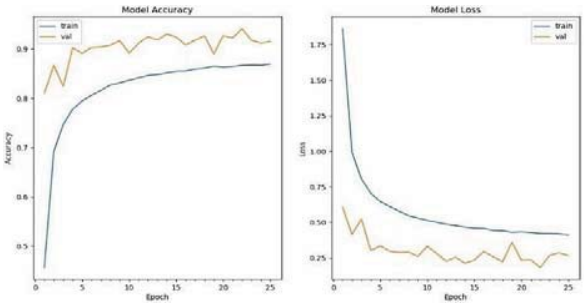

Fig. 8. Results with RMSProp Optimizer

Figure 7 and Figure 8 demonstrates the model's accuracy and model's loss after each epoch for train and validation set for Adam and RMSprop optimizer respectively.

VI. Conclusion

In this paper, we have proposed a deep convolution neural network for recognition of handwritten character in Hindi. The proposed work is basically an expansion of LeNet-5 architecture. In experiment, the model performed very well with the given datasets. The model was trained using 96000 character sets which resulted in obtaining a validation set accuracy of 95.72 per cent using Adam optimizer and 93.68 per cent using RMSprop optimizer.

There is always a scope of improvement in the machine learning models and this model can be improved in future by training more datasets and adding/improving the hyper parameter and taking care of any possible over fitting of the model. New layers may be added in between by taking care of the dimensions and maintaining a healthy bias vs variance trade-off of the model.

References

[1] L. Lu, Y. Zheng, G. Carneiro, and L. Yang, *Deep Learning and Convolutional Neural Networks for Medical Image Computing: Precision Medicine, High Performance and Large-Scale Datasets, (Advances in Computer Vision and Pattern Recognition)*, 1st ed. Springer, 2017.

[2] Devanagari Character Set. [Online]. Available: https://www.kaggle.com/rishianand/devanagari-character-set. [Accessed: 05 Nov, 2018].

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2015*, pp. 1-9.

[5] LeNet. [Online]. Available: http://yann.lecun.com/exdb/lenet/. [Accessed: 03 Jan., 2019].

[6] B. K. Verma, "Handwritten Hindi character recognition using multilayer perceptron and radial basis function neural networks," in *Proc. of IEEE International Conference on Neural Networks (ICNN'95), November, 1995*, pp. 2111-2115.

[7] M. Hanmandlu, O. R. Murthy, and V. K. Madasu, "Fuzzy Model based recognition of handwritten Hindi characters," in *Proc. of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007), December, 2007*, pp. 454-461.

[8] G. Singh and S. Lehri, "Recognition of handwritten hindi characters using backpropagation neural network," *International Journal of Computer Science and Information Technologies*, vol. 3, no. 4, pp. 4892-4895, 2012.

[9] D. Singh, M. Dutta, and S. H. Singh, "Neural network based handwritten Hindi character recognition system," in *Proc. of the 2nd Bangalore Annual Compute Conference, January, 2009*, p. 15.

[10] C. V. Jawahar, M. P. Kumar, S. R. Kiran, "A bilingual OCR for Hindi-Telugu documents and its applications," in *Proc. of the IEEE Seventh International Conference on Document Analysis and Recognition, August, 2003*, pp. 408-412.

[11] K. Verma and M. Singh, "Hindi handwritten character recognition using convolutional neural network," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 6, 2018.

[12] S. Acharya, A. K. Pant and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," in *Proc. of the 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Kathmandu, 2015*, pp. 1-6.

[13] M. Jangid and S. Srivastava, "Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods," *Journal of Imaging*, vol. 4, 2018.

[14] A. Ashiquzzaman and A. K. Tushar, "Handwritten Arabic numeral recognition using deep learning neural networks," in *Proc. of the IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Dhaka, 2017*, pp. 1-4.

[15] M. Yadav and R. Purwar, "Hindi handwritten character recognition using multiple classifiers," in *Proc. of the 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, Noida, 2017*, pp. 149-154.

[16] A. Sharma and V. Verma, "Handwritten hindi character recognition," *International Journal of Advanced Research in Electronics and Communication Engineering*, vol. 5, no. 5, 2016.

[17] J. Pradeep, E. Srinivasan and S. Himavathi, "Diagonal based feature extraction for handwritten alphabets recognition system using neural network," *International Journal of Computer Science & Information Technology*, vol. 3, no. 1, pp. 364-368, 2011.

[18] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. of the IEEE, November, 1998*, pp. 1-46.

[19] Convolution Operation. [Online]. Available: https://cdn-images-1.medium.com/max/1600/0*dRD6PhKOnnCIhz15.jpg. [Accessed: 11 Dec., 2018].

[20] Rectifier (Neural Networks). [Online]. Available: https://en.wikipedia.org/wiki/Rectifier_(neural_networks). [Accessed: 05 Dec., 2018].

[21] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.