Deep Learning Based Large Scale Handwritten Devanagari Character Recognition

Ashok Kumar Pant (M.Sc.)
Institute of Science and Technology
TU Kirtipur, Nepal
Email: ashokpant87@gmail.com

Prashnna Kumar Gyawali (B.E.)
Institute of Engineering
Pulchowk, Nepal
Email: gyawali.prasanna@gmail.com

Shailesh Acharya (B.E)
Institute of Engineering
Pulchowk, Nepal
Email: sailes437@gmail.com

***Abstract*—In this paper, we introduce a new public image dataset for Devnagari script: Devnagari Character Dataset(DCD). Our dataset consists of 92 thousand images of 46 different classes of characters of Devnagari script segmented from handwritten documents. We also explore the challenges in recognition of Devnagari characters. Along with the dataset, we also propose a deep learning architecture for recognition of those characters. Deep Convolutional Neural Network have shown superior results to traditional shallow networks in many recognition tasks. Keeping distance with the regular approach of character recognition by Deep CNN, we focus the use of Dropout and dataset increment approach to improve test accuracy. By implementing these techniques in Deep CNN, we were able to increase test accuracy by nearly 1 percent. The proposed architecture scored highest test accuracy of 98.47% on our dataset.**

***Keywords*—Handwritten character recognition, Deep convolutional neural networks, Image processing, Computer vision, Devanagari handwritten character dataset.**

I. INTRODUCTION

Character classification is an important part in many computer vision problems like Optical character recognition, license Plate recognition, etc. Development of a recognition system is an emerging need for digitizing handwritten Nepali documents that use Devnagari characters. Optical Character Recognition systems are least explored for Devnagari characters. [1] [2] present a few approaches for segmentation and recognition of Devnagari charcters. Our task is challenging because we not only have to deal with classification but also preparation of dataset. So in this paper, we introduce a new publicly available dataset, Devnagari Character Dataset, of 92 thousand images of 46 Devnagari characters. Then, we also propose Deep learning architecture to classify the characters in DCD. Introduction of multilayer perceptron network has been a milestone in many classification tasks in computer vision [3]. But, performance of such a network has always been greatly dependent on the selection of good representing features [4] [5]. Deep Neural Networks on the other hand do not require any feature to be explicitly defined, instead they work on the raw pixel data generating the best features and using it to classify the inputs into different classes [6]. Deep Neural networks consist of multiple nonlinear hidden layers and so the number of connections and trainable parameters are very large. Besides being very hard to train, such networks also require a very large set of examples to prevent overfitting. One class of DNN with comparatively smaller set of parameters and easier to train is Convolutional Neural Network [7].The ability of CNN to correctly model the input dataset can be varied by changing the number of hidden layers and the trainable parameters in each layer and they also make correct assumption on the nature of images [8]. Like a standard feed forward network, they can model complex non-linear relationship between input and output. But CNN have very few trainable parameters than a fully connected feed-forward network of same depth. CNNs introduce the concept of local receptive field, weight replication and temporal subsampling [9] which provide some degree of shift and distortion invariance. CNNs for image processing generally are formed of many convolution and sub-sampling layers between input and output layer. These layers are followed by fully connected layers thereby generating distinctive representation of the input data. Beside image recognition, CNNs have also been used for speech recognition [10] [11]. Although deep convolutional neural networks have a small and inexpensive architecture compared to standard feed forward network of same depth, training a CNN still requires a lot of computation and a large labeled dataset. Training such a network was not so effective and did not produce any superior result to traditional shallow network, until recently. With the availability of large labeled dataset like IMAGENET, NIST, SVHN, development of state of the art GPUs and introduction of unsupervised pre-training phase, CNNs have at present proven to surpass traditional feed forward network in a number of classification tasks. In CNNs, initializing the weight randomly and applying gradient descent and backpropagation to update the weights seems to generate poorer solution for a deep network [12]. So, generally, greedy layer wise unsupervised pre training is applied prior to supervised training. Why such unsupervised training helps is investigated in [13].

II. DEVANAGARI CHARACTER DATASET

A. Devanagari Script

Devanagari is part of the Brahmic family of scripts of Nepal, India, Tibet, and South-East Asia [14], [15]. The script is used to write Nepali, Hindi, Marathi and similar other languages of South and East Asia. The Nepalese writing system adopted from Devanagari script consists of 12 vowels, 36 base forms of consonant, 10 numeral characters and some special characters. Vowel characters are shown in Fig. 1, consonants characters in Fig. 2 and numeral characters in Fig. 3. Moreover, all 36 consonants could be wrapped with the vowels generating 12 other derived forms for each branch of consonant character. One such example for "*pa*" is shown in Fig. 4.

अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	अं	अः

Fig. 1: Devanagari vowels.

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	स	ष	श	ह	क्ष	त्र	ज्ञ

Fig. 2: Devanagari consonants.

०	१	२	३	४	५	६	७	८	९

Fig. 3: Devanagari numerals.

प	पा	पि	पी	पु	पू	पे	पै	पो	पौ	पं	पः

Fig. 4: Derived forms of consonant "*pa*" when wrapped with vowels.

B. Devanagari Handwritten Character Dataset

Devanagari Handwritten Character Dataset is created by collecting the variety of handwritten Devanagari characters from different individuals from diverse fields. Handwritten documents are than scanned and cropped manually for individual characters. Each character sample is 32x32 pixels and the actual character is centered within 28x28 pixels. Padding of 0 valued 2 pixels is done on all four side to make this increment in image size. the images were applied gray-scale conversion. After this the intensity of the images were inverted making the character white on the dark background. To make uniformity in the background for all the images, we suppressed the background to 0 value pixel. Each image is a gray-scale image having background value as 0.

Devanagari Handwritten Character Dataset contains total of $92,000$ images with $72,000$ images in consonant datasest and $20,000$ images in numeral dataset. Handwritten Devanagari consonant character dataset statistics is shown in Table I and handwritten Devanagari numeral character dataset statistics is shown in Table II.

C. Challenges in Devanagari Character Recognition

There are many pairs in Devnagari script, that has similar structure differentiating each with structure like dots, horizontal line etc. Some of the examples are illustrated in Fig. 5. The problem becomes more intense due to unconstrained cursive nature of writings of individuals. Two such examples are shown in Fig. 6.

TABLE I: Consonant Character Dataset

Class	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट	ठ
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000	2,000	2,000
Class	ड	ढ	ण	त	थ	द	ध	न	प	फ	ब	भ
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000	2,000	2,000
Class	म	य	र	ल	व	श	ष	स	ह	क्ष	त्र	ज्ञ
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000	2,000	2,000
Total	**72,000**											

TABLE II: Numeral Dataset

Class	०	१	२	३	४	५	६	७	८	९
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000
Total	**20,000**									

III. Character Recognition

A. Convolutional Neural Networks

Convolutional Neural Network (CNN or ConvNet) is a biologically-inspired trainable machine leaning architecture that can learn from experiences like standard multilayer neural networks. ConvNets consist of multiple layers of overlapped tiling collections of small neurons to achieve better representation of the original image. ConvNets are widely used for image and video recognition. There are three main types of layers used to build a ConvNet architecture.

1) Convolution Layer: The convolution layer is the core building block of a convolutional neural network. It convolves the input image with a set of learnable filters or weights, each producing one feature map in the output image.

2) Pooling Layer: The pooling layer is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The pooling layer takes small rectangular blocks from the convolution layer and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block.

3) Fully-Connected Layer: The fully-connected layer is used for the high-level reasoning in the neural network. It takes all neurons in the previous layer and connects it to every single neuron it has. Their activations can be computed with a matrix multiplication followed by a bias offset as a standard neural networks.

B. The Architecture

A simple convolutional neural network similar to the one used in our recognition system is shown in Fig. 7. The input layer consists of the raw pixel values from the 32x32 grayscale image and has no trainable parameters. The first convolution layer has 4 feature maps with 784 units/neurons each(28 x 28). Each feature map is shown in figure as $2D$ planes and they have different set of weights. All the units in a feature map share the same set of weights and so they are activated by the same features at different locations. This weight sharing not only provides invariance to local shift in

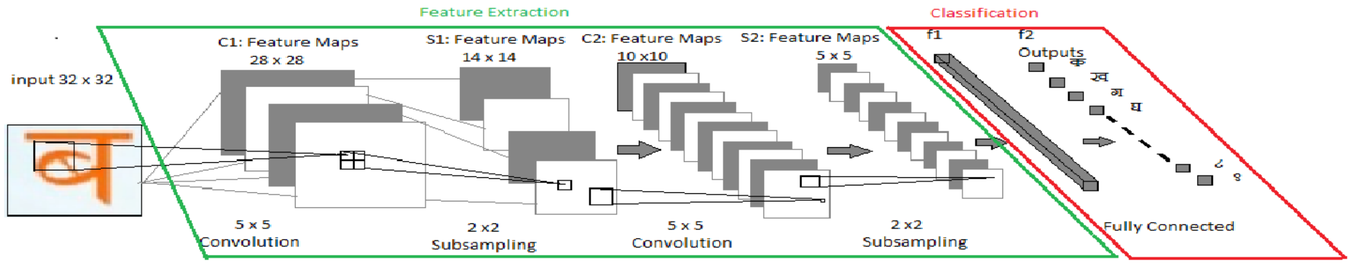


Fig. 7: Convolutional Neural Network.

छ	६	Difference being horizontal line at top
ड	ङ	Difference being presence of single dot on right side
द	ढ	Difference being presence of small circle and small down stroke line

Fig. 5: Structural formation of characters.

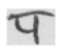
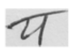
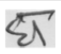
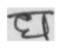
	प		य
	घ		ध

Fig. 6: Different characters written similarly.

feature position but also reduces the true number of trainable parameters at each layer. Each unit in a layer receives its input from a small neighborhood at same position of previous layer. So the number of trainable weights associated with each unit in a convolutional layer depends on the chosen size of the neighborhood of previous layer mapped to that unit. Since all the units are activated only from the input taken from a local neighborhood they detect local features such as corners, edges, end-points. This concept of local receptive field is inspired from study of the locally-sensitive, orientation-selective neurons in the cats visual system [16].

For a 5x5 kernel as shown in Fig. 7 the number of input weights for each unit is 25. In addition the units also have a trainable bias. The total number of units in a layer depends upon the size of kernel in the previous layer and overlap between the kernels.

The convolutional layer is followed by a sub-sampling/pooling layer. Sub sampling layer reduces the resolution of the feature map from convolution layer by averaging the features in the neighborhood or pooling for a maximum value. Because the exact position of features vary for different images of the same character, it is more desirable that the system does not learn the absolute position of feature but instead learn the relative position among the features. The pooling layer helps achieve this goal and makes the classifier more immune to shift and distortion. It aggregates the information within a set of small local regions and produces a pooled feature map as output. The number of units in a pooling layer thus depends upon the local region of the previous convolution layer feeding input to the units in pooling layer. So for a non overlapping scheme and a 2x2 region from previous layer connected to units in pooling layer the dimension of feature maps reduce to half of the convolution layer. The max pooling method checks for the maximum value on its local receptive field, multiplies it by a trainable coefficient, adds a trainable bias and generates output.

The second convolution layer follows this sub-sampling layer. Each feature map in $C2$ layer is generated by taking input from $S1$. The units in $C2$ get their input from the 5x5 neighborhood at identical position of some layers in S1 and not all. The reason for not connecting $C2$ feature maps to all feature maps of $S1$ layer is to reduce the total number of trainable parameters and, this also introduces randomness in providing input to different feature maps with the assumption that this will help them to learn complementary features with one another. The output of this convolution layer is subsampled, convolved and forwarded to fully connected layer. From this point we obtain a $1D$ feature vector. The fully connected layers model the input by applying non-linearity like in a traditional feed-forward network. The type of non-linearity used is ReLU-non linearity.

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

The reason for using it instead of the widely popular non-linear functions like

$$f(x) = \tanh(x) \tag{2}$$

and

$$f(x) = (1 + \exp^{-x})^{-1} \tag{3}$$

is because training with gradient-descent is comparatively much faster for ReLU than the other non-linearities [17]. The depth of the network and the size of different layers to be used depends greatly on the dataset and the problem domain. Furthermore, the number of feature maps in a layer, the size of the kernel on each layer and the choice of non-overlapping or overlapping kernel and the extent of overlap also produces different results. So, in our case we tested different architectures by varying these parameters and presented results of the architecture producing the highest accuracy on the test data set. The result of the tests are summarized on the Experimental setting and results section.

C. Over-fitting in the Deep Network

The large and deep architecture of CNN with large bank of trainable parameter makes it susceptible to overfitting. While training the deep networks, it is very difficult to find optimal hyper parameters of the functions that share the parameters. These networks being large, require large amount of training data. Below given are some approaches we used to prevent our model from overfitting.

1) Dataset augmentation: For the better recognition models, we require to have more training samples while training the system [18]. This can be achieved by augmenting available dataset by mirroring, rotation, jittering, noise injection and random crops.

2) Dropout: Dropout simply refers to dropping out units; units representing both hidden and visible in the deep network [19]. We temporarily remove the random units from the network along with all its inlet and outlet connections. For each training iterations, there will be new lighter network that remains after dropping the random units from the common denser architecture which will be sampled and trained. Each unit is retained with the fixed probability of p independent of other units and we set 0.5 for p, the number being optimal choice for most of the cases.

IV. Experiments and Result

We tested the dataset with different architectures by varying depth, width and number of parameters of network. The results of two of those experiments are presented in the coming sections. The first model is very wide and deep and consists of a large number of parameters. It will be referred to as model A in the coming section. It consists of three convolution layers and one fully connected layer. The sequence of the layers in model A is shown in Fig 2., where C is a convolution layer, R is a Rectified Linear Unit Layer, N is Normalization layer implementing Local Response Normalization, P is pooling layer implementing max pooling, D is the Dropout layer, FC is the Fully Connected Layer, A is accuracy layer for test set and SL is the Softmax Loss layer that computes multinomial logistic loss of the softmax of its input. The second model is derived from the lenet family. It has a shallower architecture and consists of fewer number of parameters than model A .It will be referred to as *model B* in the coming section. It consists of two convolutional layers followed by two fully connected layers. The sequences of layers in model B is shown in Fig 3. where each notation holds similar meaning as discussed for model A.

In all cases, Convolution is implemented with overlapping Filter(Kernel) of Size $5*5$ and stride 1 on both direction. Pooling is implemented with a non-overlapping Filter of size $2*2$ and stride 2 on both directions. Local response Normalization is achieved by dividing each input value by the expression

$$(1+(\alpha/n)\sum_i x_i^2)^\beta \qquad (4)$$

,where n is the size of each local region, and the sum is taken over the region centered at that value. The value of α-parameter used is 0.001 and β-parameter is 0.75.

Our deep neural network was trained on the DCD as a multi-class classification problem. For both the models, the standard back-propagation on feed-forward net is implemented by stochastic gradient descent(SGD) with momentum of 0.9. The mini-batch size is 200 and the network was trained for 50 epochs. The base learning rate was initialized for all trainable parameters at 0.005 for Model A and 0.001 for Model B. The learning rate was updated by an inverse function using the relation

$$LR = BLR * (1+\gamma * iterations)^{-power} \qquad (5)$$

Where BLR is the Base Learning Rate and iterations is the number iterations completed. The value of γ was set to 0.0001 and power was set to 0.75.

The result of training for 50 epoch is presented in Fig. 1. Test Accuracy remained nearly constant after 50 epochs. For modelA, Extending Dataset showed superior result in Test Accuracy. So, increasing number of training sample is effective to increase performance of wide and deep network with large bank of parameters. The highest testing accuracy obtained for Model A is 0.98471. For model B, addition of dropout showed better improvement in Test accuracy. However, extending dataset also resulted slight improvement in Test accuracy. The highest value of Testing Accuracy obtained for this model is 0.982681.

V. Conclusion

We presented a new dataset Devnagari Character Dataset which is publicly available for any researcher. It consists 92 thousand images of 46 different characters of Devnagari script. We explored the challenges in classification of characters in Devnagari Dataset. The challenges result due to the fact that the dataset consists many characters that are visually similar or written in a similar way by most people. Also, In Devnagari script, the base form of consonant characters can be combined with vowels to form additional characters which is not explored in this research. For recognition, we proposed two deep learning models to train the dataset. We also analyzed the effect of dropout layer and dataset increment to prevent overfitting of these networks. The experimental results suggested that Deep CNNs with added Dropout layer and Dataset increment technique can result in very high test accuracy even for a diverse and challenging dataset like ours.

References

[1] A. K. Pant, S. P. Panday, and S. R. Joshi, "Off-line nepali handwritten character recognition using multilayer perceptron and radial basis function neural networks," in *Internet (AH-ICI), 2012 Third Asian Himalayas International Conference on*. IEEE, 2012, pp. 1–5.

[2] V. J. Dongre and V. H. Mankar, "A review of research on devnagari character recognition," *arXiv preprint arXiv:1101.2491*, 2011.

[3] M. G. Quiles and R. A. F. Romero, "A computer vision system based on multi-layer perceptrons for controlling mobile robots," in *Proceedings of COBEM–18th International Congress of Mechanical Engineering, November,(cd-rom), Ouro Preto, Minas Gerais, Brazil*, 2005.

[4] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayer perceptron," *Journal of Neural Network Computing*, vol. 2, no. 2, pp. 40–48, 1990.

[5] J.-B. Yang, K.-Q. Shen, C.-J. Ong, and X.-P. Li, "Feature selection for mlp neural network: The use of random permutation of probabilistic outputs," *Neural Networks, IEEE Transactions on*, vol. 20, no. 12, pp. 1911–1922, 2009.

[6] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*. Citeseer, 1990.

[10] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4277–4280.

[11] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[12] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *The Journal of Machine Learning Research*, vol. 10, pp. 1–40, 2009.

[13] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[14] A. Gaur, *A history of writing*. British library, 1992.

[15] S. R. Fischer, *History of Writing*. Reaktion Books, 2004.

[16] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[17] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[18] M. Kobetski and J. Sullivan, "Apprenticeship learning: transfer of knowledge via dataset augmentation," in *Image Analysis*. Springer, 2013, pp. 432–443.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.