Deep Learning Based Large Scale Handwritten Devanagari Character Recognition

Shailesh Acharya
Institute Of Engineering
Tribhuvan University
Kathmandu, Nepal
sailes437@gmail.com

Ashok Kumar Pant
Institute Of Science and Technology
Tribhuvan University
Kathmandu, Nepal
ashokpant87@gmail.com

Prashnna Kumar Gyawali
Institute Of Engineering
Tribhuvan University
Kathmandu, Nepal
gyawali.prasanna@gmail.com

Abstract— **In this paper, we introduce a new public image dataset for Devanagari script: Devanagari Handwritten Character Dataset (DHCD). Our dataset consists of 92 thousand images of 46 different classes of characters of Devanagari script segmented from handwritten documents. We also explore the challenges in recognition of Devanagari characters. Along with the dataset, we also propose a deep learning architecture for recognition of those characters. Deep Convolutional Neural Network (CNN) 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— Devanagari Handwritten Character Dataset; Image processing; Computer Vision; Deep learning; Deep Convolutional Neural Network; Optical Character Recognition; Dropout

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 Devanagari characters. Optical Character Recognition systems are least explored for Devanagari characters. [1][2] present a few approaches for segmentation and recognition of Devanagari characters. 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, Devanagari Handwritten Character Dataset (DHCD), of 92 thousand images of 46 Devanagari characters. Then, we also propose Deep learning architecture to classify the characters in DHCD. 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 them to classify the inputs into different classes[6].

Devanagari Handwritten Character Dataset is available for download at http://www.cvresearchnepal.com/dhcd

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 Deep Neural Network with comparatively smaller set of parameters and easier to train is Convolutional Neural Network (CNN)[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 fewer 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 character dataset, Street View House Numbers dataset), development of state of the art GPU 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 back propagation to update the weights seem 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. SYSTEM OVERVIEW

The complete flow diagram of dataset preparation and character classification is presented in *Fig. 1*. Dataset preparation phase can be divided into three subsystems; character extraction, preprocessing and separation of Training and Testing set. Character extraction deals with scanning handwritten documents, cropping individual characters and labeling them. Preprocessing subsystem deals with pre-processing on the character images and the last subsystem randomly splits the dataset into training and test set. Detailed description on preprocessing and dataset preparation is presented under the subheading "DHCD Preparation". Character classification phase includes training and testing on dataset. CNN-trainer is run on the Training set and the accuracy of the trained model is tested using Testing set of DHCD.

III. DEVANAGARI HANDWRITTEN CHARACTER DATASET

A. *Script*

Devanagari is part of the Brahmic family of scripts of Nepal, India, Tibet, and South-East Asia[14]. 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. Consonant characters are shown in table 1, vowel characters in table 2 and Numeral characters in table 3.

Table 1. Base form of Consonant Characters

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

Table 2. Vowel Characters

अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	अं	अः

Table 3. Numeral Characters

०	१	२	३	४	५	६	७	८	९

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 प is shown in table 4.

Table 4. Derived forms of Consonant प when wrapped with vowels

प	पा	पि	पी	पु	पू	पे	पै	पो	पौ	पं	पः

Also there are some additional special characters in the script. Among all these characters, Devanagari Handwritten Character Dataset is for base form of 36 consonant and 10 numeral characters. It consists of total of 92 thousand images, 2 thousand images for each character.

B. *DHCD Preparation*

The 92 thousand images of DHCD were generated by imaging the characters written by many individuals resulting wide variation in the way each character is written. We scanned hundreds of handwritten documents of different writers and cropped each character manually. Each image in the dataset is unique. The Dataset is randomly split into Training (85 %) and Testing set(15 %). Training set consists of 78,200 images and the Testing set consists of remaining 13,800 images. Each image is 32x32 pixels and the actual character is centered within 28x28 pixels. Padding of 0 valued 2 pixels was done on all four sides to make this increment in image size. The cropped images were preprocessed before padding. Initially, the images were applied gray-scale conversion. After this the intensity of the pixels were inverted making the character white on dark background. To make background uniform for all the images, we suppressed the background to 0 value pixel.

C. *Challenges in DHCD*

There are many pairs of characters in Devanagari script, contained in DHCD that has similar structure differentiating each with structures like dots, horizontal line etc. Some of the examples are illustrated in table 5.

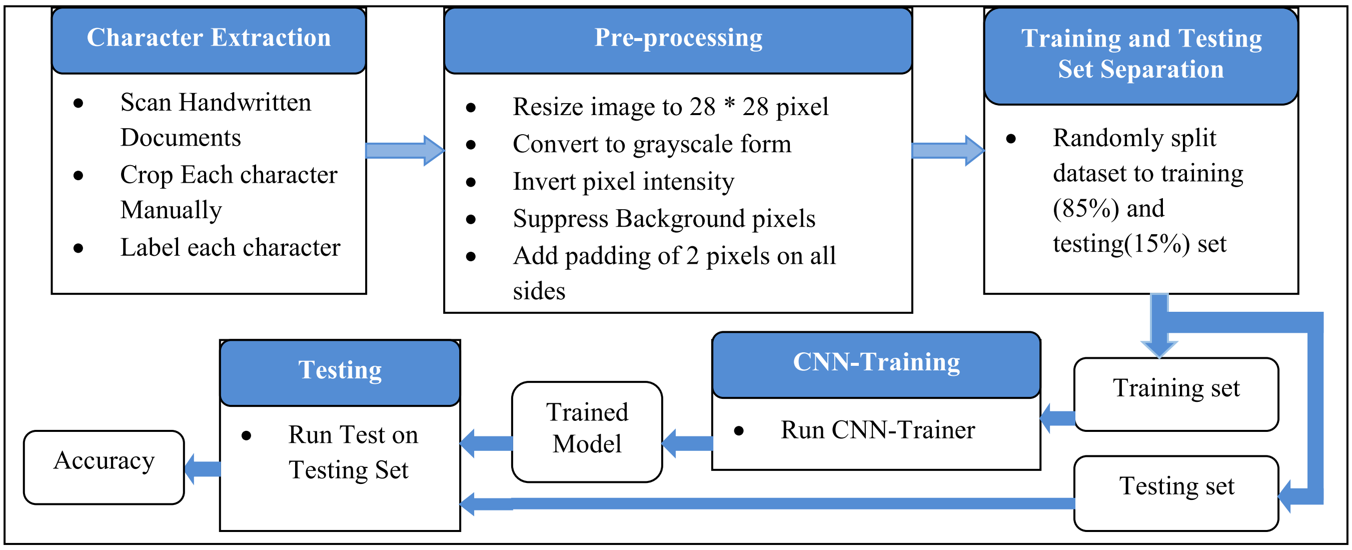

Character Extraction
Scan Handwritten Documents
Crop Each character Manually
Label each character
Pre-processing
Resize image to 28 * 28 pixel
Convert to grayscale form
Invert pixel intensity
Suppress Background pixels
Add padding of 2 pixels on all sides
Training and Testing Set Separation
Randomly split dataset to training (85%) and testing(15%) set
Training set
Testing set
CNN-Training
Run CNN-Trainer
Trained Model
Testing
Run Test on Testing Set
Accuracy

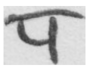
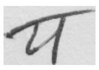
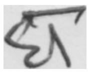
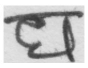
Fig. 1. General Flow Diagram of Dataset preparation and Character classification

Table 5. Structural formation of characters

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

The problem becomes more intense due to the way people write the characters. Similar scenario was seen when we collected the data for DHCD. Two such examples are shown in table 6.

Table 6. Different characters written similarly

	प		य
	घ		ध

IV. Character Recognition

A. *Convolutional Neural Network*

A simple Convolutional Neural Network similar to the one used in our recognition system is shown in *Fig* 2. The input layer consists of the raw pixel values from the 32X32 grayscale image and has no trainable parameters. The first convolution layer(C1) 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 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 by Hubel and Wiesel.

For a 5x5 kernel as shown in *Fig. 2.* 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 subsampling/pooling(S1) 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

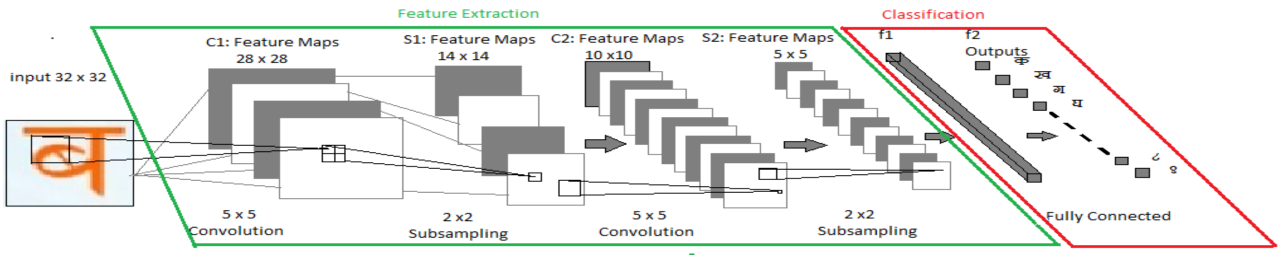


Fig. 2. Convolutional Neural Network

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(C2) follows this subsampling 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 Rectified Linear Unit (ReLU) non linearity given as $f(x) = \max(0, x)$. The reason for using it instead of the widely popular non-linear functions like $f(x) = \tanh(x)$ and $f(x) = (1 + e^{-x})^{-1}$ is because training with gradient-descent is comparatively much faster for ReLU than the other non-linearities[15].

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.

B. *Overfitting in Deep Network*

The large and deep architecture of Deep CNN with large bank of trainable parameters make it susceptible to overfitting. While training 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. The available dataset in DHCD may not be sufficient to train a network of this size. Given below are some approaches we used to prevent our model from overfitting.

1) *Dataset Increment*

During training, the number of images we trained on is much larger than the original number in DHCD Training set. Each image in training set was cropped using window of size 30x30 from the four corners and at the center generating five unique images. The position of actual character being at the center of the image in training set ensures it is contained in all five images but at different position. This technique not only increases the dataset for training but also trains the model to account for possible shift in position of character within the image. This increased dataset will be referred to as Extended set in coming section.

2) *Dropout*

Dropout simply refers to "dropping out units"; units representing both hidden and visible in the deep network. We temporarily remove the random units from the network along with all its inlet and outlet connections. For each training iteration, 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. [16]

V. EXPERIMENTAL SETTINGS 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 3.*, 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 4.* 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 following expression

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

, 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 DHCD 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 following relation

$$LearningRate = BLR \times (1 + \gamma \times iterations)^{-power} \qquad (2)$$

, where BLR is the Base Learning Rate and iterations is the number of 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. 5.* Test Accuracy remained nearly constant after 50 epochs. For Model A, Extended 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 in slight improvement in Test accuracy. The highest value of Testing Accuracy obtained for this model is 0.982681.

VI. FUTURE WORKS

Devanagari Handwritten Character Dataset is confined only to the 36 consonant characters in their base form and 10 numeral characters. However, it is essential in the future to extend the dataset to include all the vowel characters, derived form of consonant characters and special characters as well. The dataset could also be promoted to serve as a new benchmark for evaluation of handwritten character classification. The proposed classification system could be implemented to design a complete handwritten document digitizing system.

VII. SUMMARY AND CONCLUSION

We presented a new dataset Devanagari Handwritten Character Dataset which is publicly available for any researcher. It consists 92 thousand images of 46 different characters of Devanagari script. We explored the challenges in classification of characters in Devanagari Dataset. The challenges result due to the fact that the dataset consists of many characters that are visually similar or written in a similar way by most people. Also, In Devanagari script, the base form of consonant characters can be combined with vowels to form additional characters which is not explored in this research.

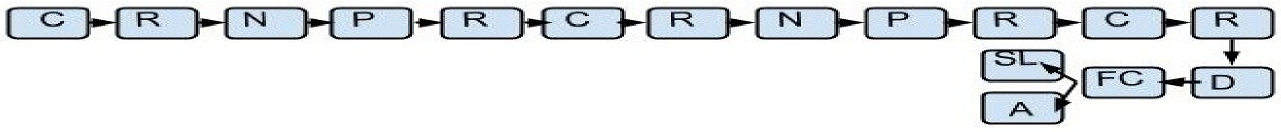


Fig. 3. Architecture of model A

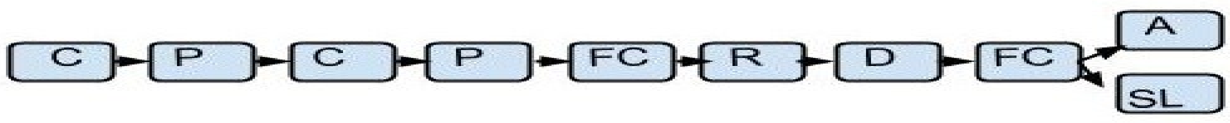


Fig. 4. Architecture of model B

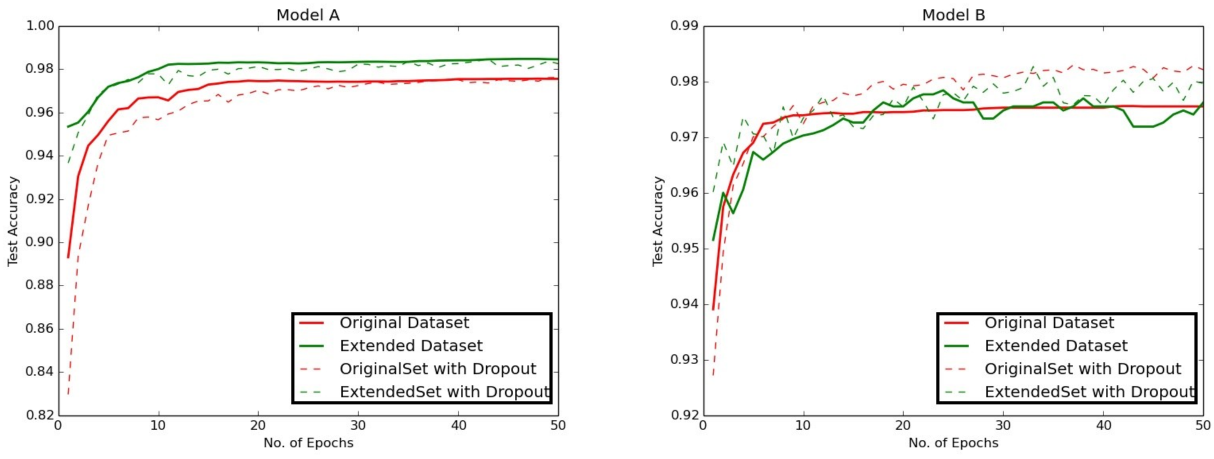


Fig. 5. Test Accuracy

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.

VIII. REFERENCES

[1] A. K. Pant, S. P. Pandey and S. P. Joshi "Off-line Nepali Handwritten Character Recognition using Multilayer Perceptron and Radial Basis Function neural networks," Third Asian Himalayas International Conference on Internet (AH-ICI), pp 1-5, 2012

[2] V. J. Dongre, V. H. Mankar "A Review of Research on Devnagari Character Recognition," International Journal of Computer Applications Vol. 12, No.2, pp. 0975 – 8887,November 2010

[3] M. G. Quiles, R. Romero, "A Computer Vision System based on Multi-Layer Perceptrons for Controlling Mobile Robots," ABCM Symposium series in Mechatronics- Vol 2 pp. 661-668

[4] D. W. Ruck, S. K. Rogers, M. Kabrisky, "Feature Selection Using a Multilayer Perceptron," Journal of Neural Network Computing, Volume 2, pp 40-48, November 1990

[5] J. Yang, K. Shen, C. Ong, X. Li, "Feature Selection for MLP Neural Network: The use of Random Permutation of Probabilistic Outputs," IEEE Transaction on Neural Networks, vol. 20,issue 12, pp. 1911-1922, October 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, pp. 609–616. ACM, 2009.

[7] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceeding of the IEEE, vol. 86, Issue 11, pp. 2278 - 2324 November 1998

[8] A. Krizebsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks,"Advances in Neural Information Processing Systems 25 (NIPS 2012)

[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Handwritten digit recognition with a back-propagation network" , Advances in Neural Information Processing Systems 2 (NIPS 1989), Denver, CO, 1990

[10] O. Abdel-Hamid, A. Mohamed, H. Jiang and G. Penn, "Applying Convolutional Neural Networks Concepts To hybrid NN-HMM Model For Speech Recognition", IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP), pp. 4277-4280, 2012

[11] T.N. Sainath, A.–R. Mohamed, B. Kingsbury and B. Ramabhadran, "Deep convolutional Neural Networks for LVCSR," IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP), pp. 8614-8618, 2013

[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. 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, Feb 2010

[14] S. R. Fischer, "A History of Writing," Reaktion Books, 2004

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," In Proc. 27th International Conference on Machine Learning, 2010

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov "Dropout: A simple way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014 .