Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 125 (2018) 525–532

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on Smart Computing and Communications, ICSCC 2017, 7-8 December 2017, Kurukshetra, India

Hybrid evolutionary approach for Devanagari handwritten numeral recognition using Convolutional Neural Network

Adarsh Trivedi*, Siddhant Srivastava, Apoorva Mishra, Anupam Shukla, Ritu Tiwari

Soft Computing and Expert System Laboratory,
ABV-Indian Institute of Information Technology, Gwalior 474015, India

Abstract

In recent years, deep learning has been extensively used in both supervised and unsupervised learning problems. Among the deep learning models, CNN has outperformed all others for object recognition task. Although CNN achieves exceptional accuracy, still a huge number of iterations and chances of getting stuck in local optima makes it computationally expensive to train. Genetic Algorithm is a metaheuristic approach inspired by the theory of natural selection and has been used for solving both bounded and unbounded optimization problems by a large success. To handle these issues, we have developed a hybrid deep learning model using Genetic Algorithm and L-BFGS method for training CNN. To test our model, we have taken the Devanagari handwritten numeral dataset. Our results show that GA assisted CNN produces better results than non-GA assisted CNN. This study concludes that evolutionary technique can be used to train CNN more efficiently.

Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications.

Keywords: Convolutional Neural Network; Genetic Algorithm; Sparse Autoencoder.

1. Introduction

Handwriting recognition (HWR) systems are models used to collect and perceive input in the form of handwritten texts from sources like paper documents and photograph. The success of handwriting recognition systems was firmly based upon the optical character recognition which is responsible for formatting and segmentation of handwritten

* Corresponding Author
Email: adarshtrivedi1996@gmail.com

1877-0509
Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications
10.1016/j.procs.2017.12.068

texts, and character recognition is the soul of this module. Character recognition is used for the identification of handwritten text to its corresponding computer documents. Several states of the art character recognition models are based on the application of deep learning model and classifiers utilizing sophisticated feature extraction methods [1].

There are more than 435 million peoples in India who uses Hindi style of writing in their daily works. Thus, there must be efficient character recognition software for the Hindi Characters which is not present yet. Therefore, with the onus to develop an effective Hindi character recognition software, we developed a model for the recognition of Hindi numerals.

The rest of the paper is divided as follows: Sect. 2 discusses the methodology of our model. Sect. 3 contains the experimental results in the form of classifier accuracy measure and other related metrics. In Sect.4, we discussed the conclusion and future works.

1.1. Convolutional Neural Network

Convolutional Neural Network (CNN) is a biologically-inspired variant of Multi-Layer Perceptron's (MLPs). The first CNN architecture was LeNet [2].

The design of the CNN can be described as the technique of shared weights or local receptive fields [3] [4]. The basic building blocks of the CNN are convolution layer, activation, and pooling. The convolution layer consists of several convolution kernels (filters) which are used to compute feature maps. In every forward pass of convolution layer, each kernel is convolved with the image to generate a feature map.

The values of feature maps then passed to activation function like ReLU and sigmoid. After activation, the features are down-sampled using pooling technique like max-pooling and mean-pooling. It partitions the image into various non-overlapping rectangles which then output the downgraded version of activated feature map.

After several rounds of convolution, activation, and pooling, the resultant images are then fed to the last layers which are a fully connected layer for the classification.

Conforming to these fundamental components, in recent years many new CNN architectures of varying complexity have come up like GoogleNet [5], ResNet [6], DenseNet[7], AlexNet [8].

1.2. Genetic Algorithm

Genetic Algorithm is a metaheuristic that mimics the natural biological genetics and belongs to a larger class of evolutionary algorithm. Genetic Algorithm was introduced by JH Holland in 1975 [9]. It can be used in solving both constrained and unconstrained problems as it searches the vast solution space efficiently. In GA, a population of candidate solutions to an optimization problem is evolved towards a better solution. In GA, there is initialization, selection, crossover, mutation and fitness evaluation as its steps. First, the initial population is generated randomly called search space, then as per their fitness, a candidate solution is generated from the original population which is most adaptable to the environment. Then, the crossover is performed on this candidate after which it is mutated, then again a new breed of the solution is selected from this population which then passed to crossover and mutation. Thus, after a specific number of crossover, mutation, and selection of solutions, the GA terminates and returns the best results.

1.3. Devanagari Numeral dataset

The Devanagari numeral database is provided by the Indian Statistical Institute (ISI), Kolkata [10]. This dataset comprises of Devanagari numerals from 0 to 9, and this dataset is considered to standard benchmark Devanagari numeral dataset, used by various authors all over the world. The dataset contains all possible handwritten numerals in Devanagari style. The Fig.1 contains few samples of handwritten Devanagari numerals from the same database. Table1 contains training sample and test sample distribution of Devanagari numerals.

Fig. 1. Handwritten Deveanagari numeral samples

Table 1. Distribution of numerals in Devanagari dataset

Digits	Training Set	Test set
0	1844	368
1	1891	378
2	1891	378
3	1882	377
4	1876	376
5	1889	378
6	1869	374
7	1869	378
8	1887	377
9	1886	378
Total	18784	3762

1.4. Related Works

In recent years, there is a rapid advancement of solutions in the object recognition and classification problems. Various models and algorithms optimization have been proposed by the researchers. Some of the classification models like Support Vector Machine (SVM) [11], Random Forest [12], K-nearest neighbor (KNN) [13] have given exceptional results on a small dataset and decent results on large datasets. But, apart from these classification models, Neural Network has outperformed these models and has been the mainstay of the solutions proposed due to the easy availability of data and its ability to give a much high accuracy and present a good generalization on the classification tasks [14].

CNN is state of the art model for the object recognition and image classification problems. The accuracy of recent works with CNN model has achieved almost near-human performance on the standard benchmark CIFAR-10, and MNIST dataset which is 96.53% [15] and 99.79% [16] respectively. CNN has been one of the most accurate models for the medical imaging and medical signal processing in ECG and EMG signals. CNN architecture has also been used in generator and discriminator for the Generative Adversarial Networks (GANs) to generate photorealistic images the purpose of visualization and to construct 3-D models of the objects from images.

Although these models have given remarkable accuracy, it also has some costs associated with it, and the costs are large computation and complex architecture development. The main reason of the costs is due to fine tuning of the weights of the neural network to extract better features and slow convergence to the optimal solution.
To remove this hurdle of slow convergence and getting stuck in local optima, many authors have given methods which uses genetic algorithm in ANN (Artificial Neural Network) weight initialization for calculating c-index in order to evaluate prognostic models for censored data [17] and for estimating the compressive strength of concrete to determine its quality [18]. GA has also been utilized to get the most efficient architecture for a given ANN [21]. In stacked autoencoder, GA has been used to get the best initial weights in each layer of the stacked autoencoder [20]. Use of GA has also been done in CNN for the weights initialization [21] [22] [23] and as well as architecture optimization [24] [25]. Many variants of GA have also been proposed in recent past [26] [27]. The effect of elite count on the behaviour of GA has been discussed in [28].
The applicability of GA depends on the encoding of the problem and fitness evaluation. And, as NN mostly used in complex and large dataset, but when weights of the NN are translated into GA (in case of weight initialization) and architecture state of the NN into GA (in case of architecture optimisation), chromosome length becomes large enough to make GA less efficient and to evaluate the fitness function, the cost of running NN becomes large which makes whole model more time-consuming.
Many authors have attempted to classify the handwritten Devanagari numeral dataset using various techniques [10]. There have been 89% accuracy achieved onto this dataset using contour extraction [29], 92% accuracy using the model based on invariant movements and division of image for recognition [30] and 95.64% accuracy using ANN + HMM [31].
In our proposed approach, we tried to classify the Devanagari numeral dataset using CNN in which the weights of the fully connected layer has been optimized using GA and the best candidate produced by the GA is then further optimized using L-BFGS algorithm. Our results indicate that when we apply second order based optimization methods on the weights that are obtained as an output of GA (instead of randomly initialized weights) results in faster convergence to the best known optimal value. We have used GA to find the best weights only for the fully connected layer of the CNN so that chromosome length remains small and computation cost is also minimal.

2. Methodology

Our proposed approach is divided into these different stages: data preprocessing, Feature extraction using sparse autoencoder, building CNN architecture, GA implementation to get best weights for the softmax classifier.

2.1. Data Pre-processing

Character normalization is considered to be the first and foremost process in the pre-processing pipeline for character recognition. The second stage in the pipeline is to bring the image onto a standard plane having fixed dimension. The goal of our pre-processing pipeline is to reduce the image noise and interclass and intraclass variation. The preprocessing task helps us to extract features of the images more easily and also improves the classification accuracy.
Our pre-processing pipeline consists of following transformations:

- Every image in our test and train dataset is converted to a greyscale image.
- For every pixel in our greyscale image, if the pixel intensity is greater than 200, then its value is changed to 255 which is white in grayscale coding.
- For every image in our dataset, each pixel value is then subtracted from a value of 0.
- Every image pixel is added by its same value, then rescaled by a factor of 1, and an offset of -100 is added.
- As the images were of a non-uniform size, every image is then converted to a uniform size of 28*28 and padding of 2 pixels is done.

Fig2. Shows the pre-processing transformations on our dataset. The first part of the image shows the raw images before the pre-processing operation. The second part of the image is after greyscaling and third part of the image is after converting the images to a uniform size and applying padding.

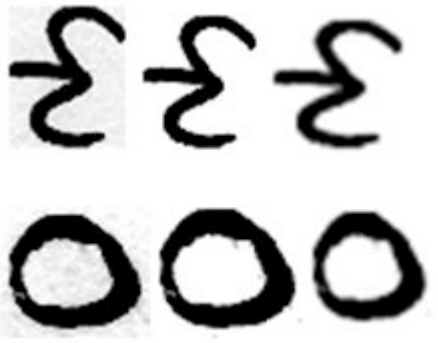

Fig. 2. Pre-processing transformation samples

2.2. Sparse Autoencoder for feature extraction

For getting the best features from our dataset, a Sparse Autoencoder was used. Autoencoder is a deep learning model which tries to learn the approximation of input data. We can extract low-dimensional features from input data by reducing the number of nodes and imposing the sparsity parameter to get essential features which can be used as kernels for our CNN classifier.To train our sparse autoencoder and get essential features from our dataset, the cost function comprising of the sum of square errors, weight decay and KL. Hyperparameters λ and β were usedfor controlling weight decay and KL distance.
In our sparse autoencoder architecture, the input layer is of dimension 1024, and the output layer is of 81. The input of our model is in vectorised images form.This sparse autoencoder is trained using L-BFGS gradient optimization for 400 iterations.

2.3. CNN architecture

Our CNN architecture consists of an input layer in which images of dimension is 32*32*3 is taken as input. These input images are then fed into convolution layer which comprises of the9*9 kernel and 256 features.
After convolution, we get the output of Dimension 256*24*24 which is then passed to the activation layer which returns the output of form 256*24*24. This activated output works as an input for the pooling layer in which we use a patch of dimension 8*8 and mean poolingis performed.
Aftermeanpooling from the pooling layer, we get the output of dimension 256*3*3 which is equal to 2304 and works as the length of the chromosome and works as an input for the Genetic Algorithm.
After running Genetic Algorithm, we get the best candidate of thesame dimension which is then passed to theinput of the Fully Connected layer which consists of a single layer softmax regressor and finally L-BFGS is also used to improve the weights of our fully connected layer further (The technical details regarding the working of L-BFGS could be referred from [32]).

2.4. Genetic Algorithm implementation

In our Genetic algorithm implementation, we first generate a population of 10 candidates; and the length of all those 10 candidates is equal to no. of weights in Softmax Classifier. The chromosomes generated follows the random normal distribution with mean = 0 and variance =1. In each iteration, 2 random candidates are chosen with equal probabilities, and their fitness is evaluated through the calculation of classification cost by initializing the weights of the classifier with the value encoded in the chromosome.
Among the 2 candidates chosen earlier, the candidate who is having the lower value of classification cost is selected. This procedure of selection is repeated 5 times to get 5 candidates for the next generation. In a different method, 5 other candidates are also generated using crossover operation on our 5 fittest candidates thus making our total

population equals 10. After that, mutation operationis carried by changing a small subset of weights to 0 with the mutation probability 0.01. This same iterative procedure is repeated and analyzed for 100, 200, and 500 generations. The candidate which we get after running this procedure for the given number of generations is then passed to Softmax Classifier for the further application ofL-BFGS gradient optimization on this.

2.5. Methodology Flowchart

Our complete methodology boils down to the flowchart in Fig4. First of all, the datasetis pre-processed by binarizing, uniform pixelation and padding of images of our dataset. Our trained dataset is then fed into the Sparse Autoencoder to generate features used for convolution and pooling, and the final pooled images are utilized by the softmax classifier for classifying the 10 digits. Genetic Algorithm has been run for the given number of generations, and the fittest candidates we get from there is passed as initial weights in Softmax Classifier.

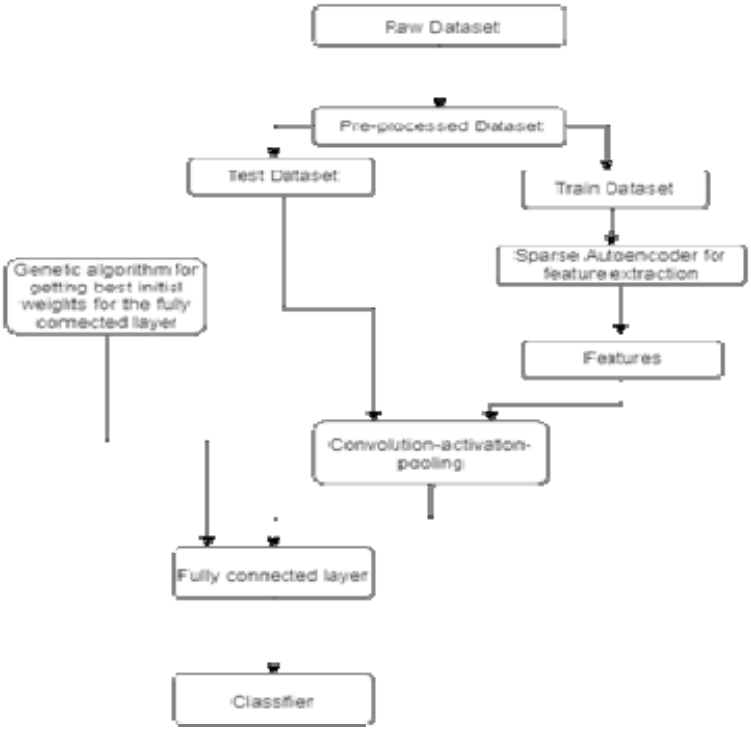

Fig. 3. Flowchart of complete methodology

3. Results

All of our simulation was done using Python3 (version 3.5) language along with sublime text editor. L-BFGS optimizer for the training of softmax classifier in CNN and sparse autoencoder was taken from Python's library Sci-Py [33]. Evaluation of our classifier was done using "metrics.classification_report" and "metrics.confusion_matrix" of Sklearn library from Python. The computer hardware which was used in our simulation consisted of 8GB of RAM, Intel Core i5 1.7 GHz processor and NVIDIA GEFORCE 820M graphics card. The time taken to run by our

sparse autoencoder to extract 256 features from our trained dataset took 4 hours of computation time. For convolution and pooling, it took 3 hours for our training dataset.

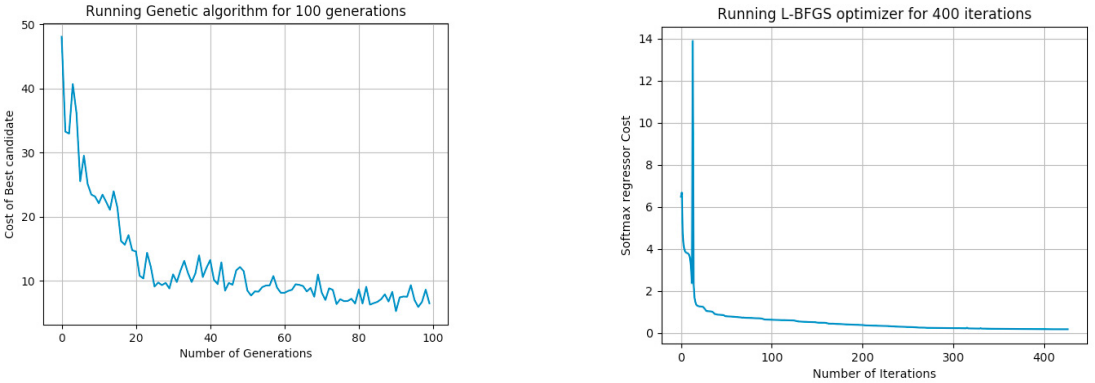

Fig4. (a)Classification cost of the best candidate for 100 generation and *(*b*)* Classification cost of training of weights in L-BFGS initialized by GA

Fig4.(a) Shows the cost of best candidates in each generation for a number of generation equals 100. The plot indicates that the cost of best candidate decreases after successive generations and the best solution after the given number of generations is given to the classifier and then these weights are further optimized using L-BFGS gradient optimization.
Fig4. (b) Shows the cost after each iteration in the training of weights by the L-BFGS gradient optimization whose weights were initialized by the GA.

Table 2. Comparison of 'CNN without GA' and with 'GA assisted CNN'

Model	Accuracy	Precision	Recall	F1-score		
CNN without using GA	96.25%	0.96		0.96	0.96	
CNN with GA (for 100 iterations)	96.41%	0.96	0.96	0.96		
CNN with GA (for 200 iterations)	96.54%		0.97		0.97	0.97
CNN with GA (for 500 iterations)	96.09%	0.96	0.96	0.96		

4. Conclusion and future works

Our experimental study shows that the by incorporating the Genetic Algorithm implementation in our Convolution Neural Network architecture improves the accuracy. The accuracy further improves a little if the number of iterations increased to 200 from 100 but decreases as we further increase the number of iterations to 500 from 200. This proves that as we are increasing the number of iterations, the length of the chromosome increases for our GA encoding technique, which makes our GA encoding technique less efficient; that's why on further increasing the no. of iterations, the accuracy drops.
For the future work, we are planning to develop a better encoding technique for our GA implementation so that length of the chromosome will not be a problem due to increase in no. of iterations. A much better architecture which we are also developing is to make GA applicable on the each layer of CNN. Thus, with the improvement in GA encoding and GA implementation on each layer, we are targeting a much higher accuracy on this dataset.

References

1. Schantz, Herbert F. (1982). The history of OCR, optical character recognition. [Manchester Center,Vt.]: Recognition Technologies Users Association. ISBN 9780943072012.
2. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278-2324. doi:10.1109/5.726791. Retrieved October 7, 2016.
3. Zhang, Wei (1988). "Shift-invariant pattern recognition neural network and its optical architecture." Proceedings of annual conference of the Japan Society of Applied Physics.
4. Zhang, Wei (1990). "Parallel distributed processing model with local space-invariant interconnections and its optical architecture." Applied Optics. 29 (32).
5. Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent; Rabinovich, Andrew (2014). "Going Deeper with Convolutions." arXiv:1409.4842
6. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). "Deep Residual Learning for Image Recognition." arXiv:1512.03385
7. Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten (2016). "Densely Connected Neural Networks." arXiv:1608.06993
8. Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton(2012). "ImageNet Classification with Deep Convolutional Neural Networks." Advances in neural information processing systems pages 1097-1105
9. J. H. Holland, Adaption in Natural and Artificial Systems. Cambridge, MA: MIT Press, 1975.
10. U. Bhattacharya and B.B. Chaudhuri, Databases for Research on Recognition of Handwritten Characters of Indian Scripts, Proc. Eighth Int"l Conf. Document Analysis and Recognition (ICDAR "05), vol. 2, pp. 789- 793, 2005.
11. Cortes, C.; Vapnik, V. (1995). "Support-vector networks." Machine Learning. 20 (3): 273-297. doi:10.1007/BF00994018.
12. Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14-16 August 1995. pp. 278-282.
13. Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression." The American Statistician. 46 (3): 175 185. doi:10.1080/00031305.1992.10475879.
14. A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained andwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.
15. Benjamin Graham (2014). "Fractional Max-pooling." arXiv:1412.6071
16. [16] Regularization of Neural Network using DropConnect Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, Rob Fergus International Conference on Machine Learning 2013
17. Kalderstam, Jonas et al. "Training artificial neural networks directly on the concordance index for censored data using genetic algorithms." Artificial intelligence in medicine 58 2 (2013): 125-32.
18. Zhe Yuan, Lin-Na Wang, and Xu Ji. 2014. Prediction of concrete compressive strength: Research on hybrid models genetic-based algorithms and ANFIS. Adv. Eng. Softw. 67 (January 2014), 156-163. DOI=http://dx.doi.org/10.1016/j.advengsoft.2013.09.004
19. Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, Babak Hodjat (2017). "Evolving Deep Neural Networks." arXiv:1703.00548
20. [20] Omid E. David and Iddo Greental. 2014. Genetic algorithms for evolving deep neural networks. In Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO Comp '14). ACM, New York, NY, USA, 1451-1452. DOI=http://dx.doi.org/10.1145/2598394.2602287
21. R. Oullette; M. Browne; K. Hirasawa (2004),"Genetic algorithm optimization of a convolutional neural network for autonomous crack detection." Evolutionary Computation, 2004. CEC2004. DOI: 10.1109/CEC.2004.1330900
22. Earnest Paul Ijjina, C. Krishna Mohan (2016). "Human action recognition using genetic algorithms and convolutional neural networks," Pattern Recognition, http://dx.doi.org/10.1016/j.patcog.2016.01.012
23. You Zhining and Pu Yunming, "The Genetic Convolutional Neural Network Model Based on Random Sample", International Journal of u- and e- Service, Science and Technology Vol.8, No. 11 (2015), pp.317-326 http://dx.doi.org/10.14257/ijunesst.2015.8.11.31.
24. Hayder M. Albeahdili, Tony Han, Naz E. Islam, "Hybrid Algorithm for the Optimization of Training Convolutional Neural Network". IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 6, No. 10, 2015
25. Lingxi Xie, Alan Yuille (2017). "Genetic CNN." arXiv:1703.01513
26. Mishra, A., & Shukla, A. (2016). Mathematical analysis of the cumulative effect of novel ternary crossover operator and mutation on probability of survival of a schema.Theoretical Computer Science, 1–11. http://doi.org/10.1016/j.tcs.2016.07.035.
27. Mishra, A., & Shukla, A. (2017). Mathematical analysis of schema survival for genetic algorithms having dual mutation. Soft Computing, 1-9. https://doi.org/10.1007/s00500-017-2650- 3.
28. Mishra, A., & Shukla, A. (2017). "Analysis of the Effect of Elite Count on the Behavior of Genetic Algorithms: A Perspective", 7th IEEE International Advance Computing Conference (IACC-2017), VNR VignanaJyothi Institute of Engineering and Technology, Hyderabad, India, January 2017. 10.1109/IACC.2017.0172
29. G S Lehal, Nivedan Bhatt, "A Recognition System for Devnagri and English Handwritten Numerals," Proc. Of ICMI, 2000.
30. R.J.Ramteke, S.C.Mehrotra, "Recognition Hand-written Devanagari Numerals," International Journal of Computer Processing of Oriental Languages, 2008.
31. U. Bhattacharya, S. K. Parui, B. Shaw, K. Bhattacharya, Neural Combination of ANN and HMM for Handwritten Devanagari Numeral Recognition.
32. Byrd, R. H., Lu, P., Nocedal, J. and Zhu, C.: 1995, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16(5), 1190–1208.
33. Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37