Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 2, No. 1, February : 2023

IMPROVING PERFORMANCE OF DEEP NETWORKS ON HANDWRITTEN DIGIT CLASSIFICATION

Dr. Saduf Afzal, Dept. of computer sciences, University of Kashmir, J&K, India
Dr. Shifaa Basharat, Dept. of computer sciences, University of Kashmir,J&K, India,
Dr. Shozab Khurshid, Dept. of computer sciences, University of Kashmir,J&K, India
M. Arif Wani, Dept. of computer sciences, University of Kashmir, J&K, India
sadafwant@gmail.com; shifa_csct@uok.edu.in ;shozabkhurshid@gmail.com
awani@uok.edu.in

Abstract
Recently, there has been surge in the use of models composed of several layers of nonlinear processing. It has been established that deep networks are more suitable in tackling complex problems on large datasets. Unsupervised layer-wise pre-training and subsequent supervised fine tuning has shown promising results in the training of deep networks. Unsupervised pre-training which involves stacking of restricted Boltzmann machine is used to initialize the parameters of deep network, while as supervised fine tuning improves upon what has been learned in the pre-training stage. The most commonly used algorithm in the supervised fine tuning stage is back- propagation. In this paper we explore the performance of dropout and No-prop in supervised fine tuning of deep networks. The performance of these algorithms is evaluated on MNIST dataset. Experimental results demonstrate that dropout gives better results in training of deep networks.

Keywords: back-propagation; deep neural network; dropout; No-Prop; restricted Boltzmann machine.

I. INTRODUCTION

MNIST handwritten digit recognition problem [1] is a well-known benchmark problem used extensively in the field of machine learning. Artificial neural networks (ANN) with single layer of nonlinear processing (shallow architecture) were among the first models evaluated on MNIST. However ANN with single hidden layer did not yield promising results on MNIST. Recent developments have demonstrated that deep architectures can represent highly varying functions much more efficiently and compactly (sometimes exponentially) than shallow architectures, in terms of number of computational elements and parameters [2]. However, training deep neural networks (DNN) has proven to be challenging. The standard strategy for training neural networks which consists of randomly initializing the parameters of neural network followed by training using back-propagation (BP) algorithm, did not work well in the context of DNN. It was empirically observed that training DNN using BP algorithm results in finding poor solutions and sometimes display performance worse than shallow networks. The reason behind the poor performance of the DNN has been attributed to the gradient based nature of BP algorithm. The BP algorithm is likely to get stuck in poor local minima. The number and the width of these local minima increase with increase in the number of layers, which leads to sub optimal results. Furthermore the gradient information vanishes as it is back propagated from layer to layer [3]. Another disadvantage of BP algorithm is its slow learning speed. It becomes computationally excruciating when the depth of network increases, as it requires propagating errors across the network. Moreover BP algorithm requires the presence of labelled data, which entails a problem when labelled data is scarce and non-existent.

Many novel strategies were introduced to alleviate the problem of training deep DNN. In this paper our objective is to explore the use of No-Prop [4] and dropout [5] in the training of DNN. An empirical analysis is done to investigate the performance of these algorithms in fine tuning of DNN that have been initialized using the unsupervised pre-training. The performance of these algorithms is evaluated on the MNIST dataset. The goal of this study is to determine which of the algorithm yields substantial improvement in training of DNN on MNIST dataset.

The paper is structured as follows. Section II details some recent works done to overcome the problem of training DNN. Section III describes the deep neural network architecture that we have used in detail. Section IV describes the dropout and No-prop. In Section V we present the experimental results. Finally, the conclusion is drawn in Section VI.

II. LITERATURE REVIEW

A breakthrough in training of deep architectures happened with the proposal of greedy layer-wise unsupervised pre-training [6], as a way of initializing deep supervised neural network. It was empirically observed that unsupervised pre-training often yields better representations [8] [7]. A supervised version of layer-wise pre-training called the greedy layer-wise supervised pre-training was given by [9]. The performance improvements obtained using [9] were not comparable to those obtained using unsupervised pre-training; nevertheless they were far better than using no pre-training at all. Reference [10] used the previous layer outputs as extra inputs to the subsequent layer (in addition to raw input). Another variant [11] pre-trains in a supervised way all the previously added layers at each step of the iteration.

Based on the idea that the flow of activations and back-propagated gradients should be maintained in a deep architecture, well designed initialization strategies were put forward for training DNN. Reference [12] proposed a new weight initialization scheme that significantly improved the convergence speed of deep networks. This technique preserved the variance of activation and back-propagated gradients. Other similarly motivated strategies were proposed by [13] [14]. Reference [14] introduced the concept of setting initial conditions on the weights, while as [13] presented a new heuristic technique for initializing DNN, a procedure called as Random Walk Initialization.

Recent studies have established that the choice of activation functions have a strong influence on the training of deep architectures [12]. Reference [15] explored the use of rectifier neurons in the training of deep networks. It was established that this hard-limiting non-linearity actually works very well and can show best performance without using unsupervised pre-training for parameter initialization when large labelled datasets are existent. Furthermore, it expedites convergence of the training procedure [16] and leads to remarkable results [15]; than conventional sigmoid like units. Reference [17] proposed a generalized version of rectified unit called Parametric Rectified Linear Unit (PReLU). PReLU improved model fitting without requiring any extra computational cost and minuscule risk of over-fitting. Activation functions [18] [19] based on the local competition were also put forward to train DNN.

III. DEEP NEURAL NETWORK

A deep neural network (DNN) is a multi-layer feed-forward, artificial neural network. DNN takes input from the lower layers, processes it, until the upper layers learn a higher level representation of the data. Lower layers produce features which are combined to produce higher level features at the next level. However, training them using standard procedure is known to be hard. Unsupervised pre-training using stacked RBM followed by fine-tuning is known to yield significant performance improvement over fine tuning using random initialization. Unsupervised pre-training provides initial configuration to the parameters with which gradient based training is initialized. In this paper, instead of initializing the parameters of DNN randomly; we have used unsupervised pre-training for the parameter initialization. The DNN is constructed in the following manner: The initial RBM is trained on the input data by contrastive divergence (CD) algorithm; the outputs of the trained RBM create a second level representation of data which is used to train the second RBM. This process is iterated until a predefined number of RBM's is trained. This stack of RBM's is called Deep Belief Network (DBN). The trained DBN can be used as a generative or discriminative model. In this paper we have used DBN as discriminative model by adding to it an output layer. We refer to the constructed network as DBN-DNN. Subsequently, the parameters (weights and biases) of DBN-DNN are then fine-tuned in supervised manner for better discrimination. The supervised algorithms that we have used for the fine tuning of the constructed model are discussed in Section IV. The unsupervised training of RBM's is referred as pre-training and the normal supervised training of the whole network is called fine tuning. The unsupervised pre-training step finds features in the training data and fine tuning phase improves the recognition ability by slightly modifying "these features to adjust the boundaries between classes" [20].Unsupervised pre-training initializes the parameters of deep networks to sensible values so that they represent the structure of input data in a more meaningful way than the random initialization, thereby yielding significant improvement in the generalization performance [2].

Following sections describe the components of DBN-DNN:

A. *Restricted Boltzmann Machine*

RBM is a generative model that is able to generate samples similar to those in the training set. The training procedure of RBM involves adjusting the weights in such a manner so as to maximise the probability of generating the training data. RBM consists of two layers of neurons: visible layer and hidden layer. All the neurons in the visible layer are connected to the neurons in the hidden layer, with no intralayer connections [2].

The probability distributions over couples (v , h) of visible units and hidden units is given as:

$$p(v, h; W, c, b)=\frac{1}{Z} e^{-E(v,h;W,c,b)} \quad (1)$$

Where z is the partition function; E is the energy function; W represents weights; b is the bias for hidden units and c is the bias for visible units.

The Energy function E is

$$E(v, h; W, c, b)=-c^T v - b^T h - v^T wh \quad (2)$$

The states of the visible vector v are associated to the input data and the hidden vector h presents the states of the hidden neurons, i.e. the hidden features. Given a data vector v, the conditional probability that the units of the hidden layer will be activated can be given as:

$$P(h_j = 1|v) = sigm\left(b_j + \sum_{i=1}^{m} w_{ij} v_i\right) \quad (3)$$

Where $sigm = {}^{1}/_{1\,+\,exp^{-x}}$ is the sigmoid activation.

The hidden states can then be used to produce the reconstruction of the data by activating the units in visible layer with conditional probability:

$$P(v_i = 1|h) = sigm(c_i + \sum_{j=1}^{n} w_{ij} h_j) \quad (4)$$

B. Contrastive divergence

RBM's are trained to improve the reconstruction ability and thus to maximize the log- likelihood of training datalog $p(v^{(0)})$ over the training parameters for a given training example $v^{(0)}$[6]. In order to train the parameters of RBM (W, b, c), a training algorithm –contrastive divergence (CD) [21] is used. The parameters of RBM are updated using CD as:

$$\Delta w_{ij} = \eta < v_i^0 h_j^0 >_{data} -< v_i^1 h_j^1 >_{model} \quad (5)$$

where η is the learning rate; $<.>_{data}$and $<.>_{model}$are the expectations under the distributions defined by the data and the model. Given the training data v^0, the hidden states can be inferred from (3) and in a similar manner visible states can be inferred from hidden states using (4).

IV. DROPOUT AND NO-PROP

For the supervised fine tuning of pre-trained deep networks, a number of supervised learning algorithms are available. However, BP is known to yield remarkable results in the fine tuning of deep networks. BP algorithm propagates the error across the neural network to adjust the weights. Training of neural network using BP involves two stages: feed-forward and back-propagation stage. In the feed forward stage, training patterns are presented to the network and the output is calculated. The output calculated is compared with the target output and an error is computed. In the back-propagation stage, the error calculated is propagated backwards to change the weights of network. This procedure is iterated until a particular termination criterion is satisfied. Here we experiment with the recently introduced techniques used in the training of neural networks i.e. Dropout and No-Prop. Both of these techniques are used in the supervised fine tuning of the pre-trained networks and their performance is compared with the fine tuning done by standard BP.

Dropout [5] is a recently introduced technique for training deep networks. It can be interpreted as a special way of averaging different models that share subset of parameters. It involves probabilistically dropping out neurons from the network on each presentation of training example. Activations flow only through those neurons that are retained and in a similar manner errors are back propagated only through the non-dropped neurons. When any neuron is dropped it is equivalent to dropping all its weighted connections. The combination of dropped out neurons is unique for a large network, thus each input vector trains its own network, which excessively share the retained weights with other networks. The neurons in the dropout are dropped out randomly with a probability $q = 1 - p$. The main objective of dropping out neurons is to prevent coadaptation in neurons, such that each neuron behaves as a reasonable model without relying on other neurons being there. This procedure forces each neuron to be more robust, independently useful and pushes it towards creating more meaningful representation instead of relying on others. Using dropout is equivalent to sampling a sub-network from it. The sub-network consists of a combination of neurons that endure the dropout. If a neural network consists of n units, we can have 2^n possible sub-networks. Given a training vector, a new sub - network is produced and trained. Under this interpretation, a neural network trained using dropout can be seen as training a collection of 2^n sub-networks. After the training of the sub-networks is finished, the 2^n sub-networks that involve excessive weight sharing are combined into a single network. This single network is then used at the test time; however the predictions at the test time are produced by suitably scaling the weights. The idea is that if at the time of training, a layer is trained with dropout and all the units of that layer are retained with a probability p. Then at the prediction time the output weights of these units must be multiplied by p.

Consider a fully connected feed forward neural network with H hidden layers. Let $h \in \{1, \ldots, H\}$be the hidden layers of the network. Let $\boldsymbol{s}^{(h)}$be the input vector into layer h and $\boldsymbol{o}^{(h)}$ be the output vector from layer (By convention, we define $o^{(0)} = \mathrm{x}$ is the input). Let $W^{(h)}$ be the matrix of weights and $b^{(h)}$vector of bias terms at layer h .The forward flow of activation in standard neural network (for hidden unit i) can be given as:

$$s_i^{(h+1)} = \boldsymbol{w}_i^{(h+1)}\boldsymbol{o}^{(h)} + b_i^{(h+1)} \tag{6}$$

$$o_i^{(h+1)} = g\left(s_i^{(h+1)}\right) \tag{7}$$

The forward flow of activation using dropout procedure can be described as:

$$\widetilde{\boldsymbol{o}}^{(h)} = m^{(h)} * \boldsymbol{o}^{(h)} \tag{8}$$

$$s_i^{(h+1)} = \boldsymbol{w}_i^{(h+1)}\widetilde{\boldsymbol{o}}^{(h)} + b_i^{(h+1)} \tag{9}$$

$$o_i^{(h+1)} = g\left(s_i^{(h+1)}\right) \tag{10}$$

where g is any activation function and $m^{(h)}$ is drawn independently from Bernoulli distribution.

No-prop [4] is a simple and effective algorithm for training feed forward networks. No-Prop stands for no back-propagation. This algorithm involves setting the weights of hidden layer neurons to some random values and training only the output layer weights. The network output weights are trained using the steepest descent and the objective is to minimize the error using the LMS gradient algorithm of Widrow and Hoff. Instead of back-propagating the errors across the entire network, it involves adapting only the network output weights with the weights of hidden layers being set and fixed with random values. No-Prop algorithm always converges on a global optimum and as the error is propagated only to the last hidden layer; the problem of gradient vanishing becomes less evident, which occurs in case of BP. However, for training over capacity networks, No-Prop requires last hidden layer to be much larger, than neural networks trained using BP.

V. EXPERIMENT AND RESULTS

In order to measure the benefit of dropout and No-prop in the training of deep neural networks, we have used them in the fine tuning of deep network which have been pre-trained using unsupervised feature learning. The performance of these algorithms is compared with standard BP. All of these methods are evaluated on MNIST dataset of handwritten digit images. The MNIST dataset consists of 70,000 labelled handwritten digit images, each representing a 28×28 pixel grayscale image of one of the 10 digits(0 – 9). The dataset is partitioned into training set of 60,000 and test set of 10,000 images. The selected architecture of the neural network is 784-1200-1200-10 for DBN-DNN fine-tuned with dropout and BP. However for No-Prop we have used 784-1200-2000-10 architecture as we are training over Capacity network, which requires increasing the neurons in the last hidden layer. In this paper dropout is used at the input layer as well as at the hidden layer. At the input layer, the input components are retained with the probability of 0.8. While as, at the hidden layer the units were retained with probability of 0.5.

The performance is evaluated using the following metric:

$$err = {}^{N_{inc}}/_{N} \qquad (11)$$

where N_{inc} is the count of miss classified inferences and N stands for the number of training samples.

The comparison results are summarized in Table I. The experimental results demonstrate that fine tuning with dropout gives better results in comparison to No-Prop and BP.

Table 1: Performance comparison on mnist

Method	Error[%]
DBN-DNN+ BP fine tuning	1.68
DBN-DNN + Dropout fine tuning	1.57
DBN-DNN+ No-Prop fine tuning	9.59

VI. CONCLUSION

Training deep architectures has proven to be challenging. Recent developments have demonstrated the use of layer-wise unsupervised pre-training as an effective way of improving the performance of deep neural networks. This work explores the use of No-Prop and dropout in the supervised fine tuning of deep neural networks, the parameters of which have been initialized using unsupervised pre-training. Our simulation results show that fine tuning by dropout gives better results in comparison to No-Prop. The improvements that we obtained using dropout suggest that fine tuning is a crucial stage in the training of deep networks and moving beyond the traditional way of fine tuning deep networks is worthy of the further research. For future work, it may be promising to consider the fine tuning of deep networks using different variants of BP combined with the dropout strategy.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, pp. 2278–2324. 1998.
[2] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, “Exploring strategies for training deep neural networks,” Journal of Machine Learning Research, vol. 10, pp. 1–40, January 2009.
[3] D. Erhan, P.A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, “The difficulty of training deep architectures and the effect of unsupervised pre-training, ” AISTATS, pp. 153–160, 2009.
[4] B. Widrow, A. Greenblatt, Y. Kim, D. Park, “The No-Prop algorithm: A new learning algorithm for multilayer neural networks,” Neural Networks, vol. 37, pp. 182–188, 2013.

[5] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15(1),pp. 1929-1958, 2014.

[6] G. E. Hinton, S. Osindero and Y. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, pp. 1527–1554, 2006

[7] R. Salakhutdinov and G.E. Hinton, "Deep Boltzmann Machines," AISTATS, pp. 448-455, 2009.

[8] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, and S. Bengio, "Why Does Unsupervised Pre-Training Help Deep Learning?," Journal of Machine Learning Research, vol. 11, pp. 625-660, 2010.

[9] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," Advances in Neural Information Processing Systems, pp. 153-160, 2007.

[10] D. Yu, S. Wang, and L. Deng, "Sequential labeling using deep- structured conditional random fields," IEEE Journal of Selected Topics in Signal Processing, vol. 4, pp. 965-973, December 2010.

[11] F. Seide, G. Li, and D. Yu, "Feature engineering in context- dependent deep neural networks for conversational speech transcription," 2011 IEEE Workshop Automatic Speech Recognition and Understanding, ASRU 2011, pp. 24-29, 2011.

[12] X. Glorot and Y Bengio, "Understanding the difficulty of training deep feedforward neural networks," Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, vol. 9, pp. 249–256, May 2010.

[13] D. Sussillo, and L. F. Abbott, "Random walk initialization for training very deep feedforward networks," arXiv preprint arXiv: 1412.6558, 2014.

[14] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," arXiv preprint arXiv: 1312.6120, December 2013.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, AISTATS' 11, vol. 15, pp. 315-323, 2011.

[16] K. Alex, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," Neural Information Processing System, pp. 1097-1105, 2012.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," Proceedings IEEE International Conference on Computer Vision., pp. 1026-1034, February 2016.

[18] J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio, "Maxout networks." Proceedings of the 30th International Conference on Machine Learning (ICML), vol. 28, pp. 1319-1327, February 2013.

[19] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber,"Compete to compute," Advances in Neural Information Processing Systems, pp. 2310-2318, 2013.

[20] G. E. Hinton, "Learning multiple layers of representation," Trends in Cognitive Sciences, vol. 11, pp. 428–434, 2007.

[21] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," Neural Computation, vol. 14, pp. 1771–1800, 2002.