International Journal of Database Theory and Application
Vol.10, No.8 (2017), pp.21-30
http://dx.doi.org/10.14257/ijdta.2017.10.8.03

Weight Initialization based Partial Training Algorithm for Fast Learning in Neural Network

Jung-Jae Kim[1], Min-woo Ryu[2], Si-Ho Cha[3] and Kuk-Hyun Cho[1]

[1]Department of Computer Science, Kwangwoon University,
447-1, Wolgye-dong, Nowon-gu, Seoul 139-701, South Korea
[2]Service Laboratory, Institute of Convergence Technology
KT R&D Center, 151, Taebong-ro, Seocho-gu, Seoul, South Korea
[3]Dept. of Multimedia Science, Chungwoon University
113, Sukgol-ro, Nam-gu, Incheon, South Korea
[1]{kjj6929, chokh}@kw.ac.kr, [2]mw.ryu@kt.com, [3]shcha@chungwoon.ac.kr

Abstract

The classification problem is one of most important problems in Artificial Intelligence (AI) Research. Classification is used in various fields such as speech recognition, image classification, word prediction in text. Deep Neural Network (DNN) is the most commonly used for the classification. However, DNN requires a lot of learning time because of its deep network structure and lots of data. At this time, if a new feature or a new category class (new data) is added, the existing data on which learning has been completed is also re-learned. And the same learning time (very long time) as the previous learning time is needed. Therefore, in this paper, we proposes Weight Initialization-based Partial Training (WIPT) algorithm, that decompose the existing weight matrix through Singular Value Decomposition (SVD) and generate a latent matrix with information learned by the existing model. In order to increase the learning efficiency, we use a strategy of learning new features or classes by initializing newly added weights to appropriate values. Finally we verify the efficiency of the proposed algorithm by comparing it with the existing whole learning.

Keywords*: We would like to encourage you to list your keywords in this section*

1. Introduction

The classification problem is one of the most important problems in Artificial Intelligence (AI) Research [1]. In a real environment, classification problems have a large impact on human activities. In order to perceive contexts in contextually or make context-sensitive decisions, humans first classify the current situation using available information [2]. In artificial intelligence, the way a machine determines behavior for a situation is similar to human's behavior. AI first classifies which category of data belongs to a given attribute or feature value. Thereafter, the actions that AI can take depends on the class in which the data is classified.

Classification is used in various fields such as speech recognition, image classification, word prediction in text information. In order to solve this classification problem, pattern recognition is important for patterning given information and matching patterns to categories. Typical pattern recognition is generalizing the information given through a specific formula and making it a pattern, and it has been difficult to generalize the feature values [3]. However, recent development of Deep Neural Networks (DNN) has helped solve the problem of pattern recognition.

DNN consists of a deep layer of basic neural networks. In DNN, pattern recognition is a high-level abstraction of data on feature values as they pass through several layers, and it means finding a representative pattern representing data through abstraction [4]. These

ISSN: 2005-4270 IJDTA

characteristics led to an image recognition competition called ImageNet Challenge, which showed a high performance of 97 percent [5-7]. In speech recognition, especially Large Vocabulary Continuous SR (LVCSR) task, the performance is higher than that of existing machine learning based pattern recognition [8-10].

DNN guarantees high performance in the classification problem through pattern recognition. In general, in order for the DNN model to have high accuracy, it is necessary to acquire a lot of training data and increase the DNN layer depth and network size. At this time, time required for learning increases due to large data. And the increase of the network size exponentially increases the number of weight parameters inside the DNN and the learning time also increases by increasing the number of weight parameters. So, if a new feature or a new category class is added, the existing data on which learning has been completed is also re-learned, and re-learning time equivalent to the existing learning time is required.

Therefore, in this paper, we propose a Weight Initialization based Partial Training (WIPT) algorithm that initializes newly added weights to learn only newly added features and parameters in order to improve the temporal efficiency of learning. WIPT algorithm initializes the weight parameters for newly added data to appropriate values, and learns only newly added data and converges to the optimal point within a short time. Therefore, we decompose the existing learned weight matrix through Singular Value Decomposition (SVD) and generate a new weight matrix to make a latent matrix containing information learned by the existing model.

The rest of this paper is organized as follows. The existing related work in this research field is introduced in Section 2. The WIPT algorithm is presented in Section 3. And Section 5 presents a performance evaluation between WIPT algorithm and whole learning. Finally, we conclude our remark in Section 5.

2. Related Work

2.1. The Progress of Neural Network

Figure 1. Shows a typical Feed-Forward Neural Network (FNN). FNN consists of three parts: input layer, hidden layer, and output layer. The input layer is responsible for transferring learning data to the network. The hidden layer is found representative pattern by abstracting the input data. Finally, the output layer shows the result of classifying the input data. All layers are fully connected with the weighted parameters and learning in the neural network means adjusting the weight parameters and learning in the neural network means adjusting the weight parameters to reduce the difference between the output and the actual result through the model.

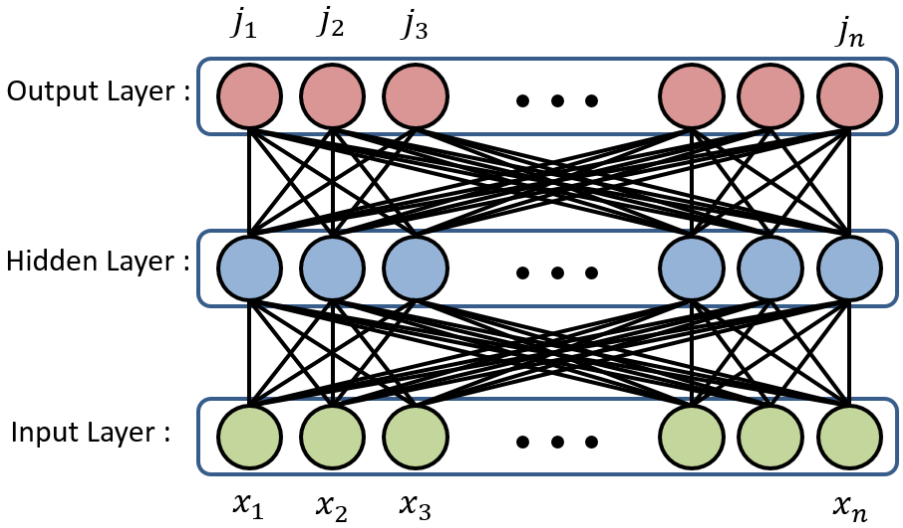

Figure 1. Typical Structure of Neural Network

2.2. Recent Research of Fast Learning

In DNN, the research on fast learning can be classified into two methods: fast convergence to optimal point and meta learning strategy. Recent research has been conducted to prevent overfitting by converging through a small number of iterative learning operations (epoch).

First at all, the optimization method is an approach to adaptively adjust a fixed learning rate. This objective is to change learning rate largely away from the optimal point and to change the learning rate to a small value near the optimal point, so that it can converge to the optimal point with a small epoch. J. Duchi poposed an algorithm called Adagrad [11]. The poposed algorithm continuously keeps track of the squared value of the gradient vector and performs a kind of standardization function for each weight at the time of parameter update through the slope vector. At this time, the weight parameters having a high slope value decrease the effective learning rate and the weight parameters having a low slope value or less updating value increase the real learning speed. G. Hinton found that the Adagrad algorithm has a disadvantage that the weight parameter increases or decreases radically in one direction or stops learning too quickly because the learning is monotonic, and RMSprop [12] is proposed. RMSprop showed that the problem can be solved by change Adagrad algorithm to a moving average of the gradient vector. Thus, the actual learning speed is similar to that of Adagrad, but there is no monotonically decreasing in learning rate. D.Kingma recently proposed the Adam algorithm [13]. The Adam algorithm works as if we added a momentum term to RMSprop to speed up the learning rate of RMSprop. This leads to a better convergence speed than the RMSprop algorithm on average. In the case of this optimization methodology, since the initial value of the weight is greatly affected, it takes a considerable time to converge the optimal point when the initial value is biased.

And seconds, the meta learning strategy is a subroutine integrated into other algorithms to complement the optimization method. X.Glorot [14] found that learning using a random initialization method in the DNN using the sigmoid activation function does not work because of the average value, and he proposed the Xavier initialization method. Xavier initialization improves the convergence speed of neural networks by using a method of distributing the neurons of one layer to approximately the same output distribution through the dispersion correction method. This can prevent bias of weights that may occur through initialization by assuming that the output of the neural network to the input data has the same identically distribution. After that K. He [15] proposed He's

initialization, which is a weight initialization method in a neuron unit using the ReLu activation function. He's initialization allowed the asymmetry of the neurons using the ReLu activation function by fixing the variance of the neurons to 2/n

Recently, many studies have been conducted to accelerate the learning speed of deep learning [11-15]. However, for partial learning of newly added data, it is importanct not only to quickly converge to the optimal point but also to set an initial value for the weight parameters to be added appropriately.

3. SVD based Weight Initialization

This chapter describes what WIPT algorithm proposed in this paper. And this chapter consists of four parts and describes the motivation for the WIPT and defines the cost function for parameter initialization. Then, the update rule for the gradient descent method is drived based on the defined cost function. Finally, the procedure of the WIPT algorithm is described.

3.1. Motivation

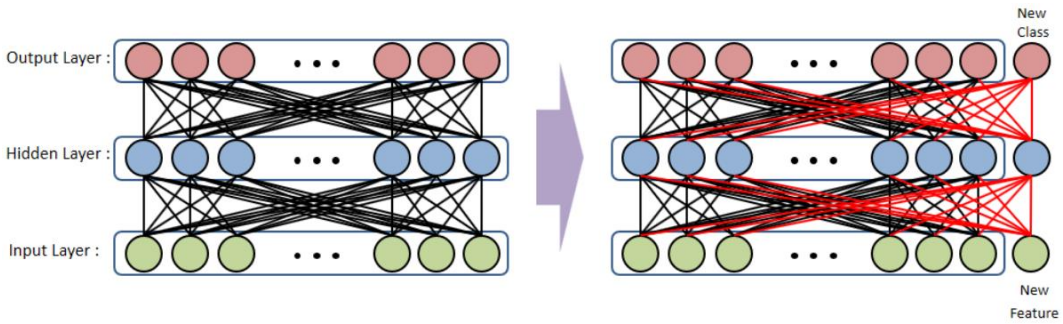

Figure 2. Concept of Weight Initialization

G. E. Hinton found that well-defined parameters in DNN shows very good learning efficiency. It is important to initialize weight parameters through the study of Hinton, and it can be assumed that weight parameters of DNN that has been learned already has information on the previous training data. The idea of this paper start from the above assumption. If a new feature is added, new weight parameters are created that is linked to the neuron as shown in the Figure 2. If the new weight parameters is generated based on the already learned parameter instead of the random initialization, it will contain the existing characteristic and it will be able to converge quickly to the optimal point even if only the newly added data is learned. If we simplify problem to obtain added new edges, the problem is to extend new matrix of m x (n +1) as in Figure 2 when the existing matrix X of adjacent layer is m x n. However, at that time we have restrictions for example if a learned neural network model was well classified training data, the learned weight matrices already have information regarding training data. Accordingly, the new matrices should have same characteristics as existing matrix X as a latent matrix. To do this end, we use Singular Value Decomposition (SVD) [3] which used to extract singular values and also estimate similar based on cost function.

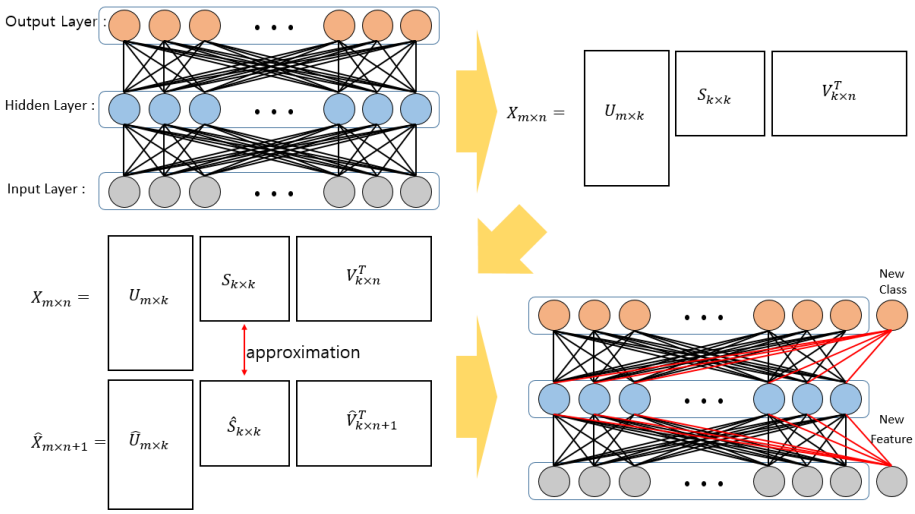

Figure 3. Concept of Weight Initialization

Therefore, the procedure to initialize the newly created weight parameters is shown in Figure 3. First, the existing learned weight parameters are decomposed using SVD. Then, the matrix to be newly generated is decomposed using SVD, and then the diagonal matrices of the two matrices are compared. The cost function then reduces the difference between two diagonal matrices, so, that information on the data of the existing weight parameters are also given a new weight parameters. Thereafter, weight parameter initialization including information on existing data is possible by copying only the portion of the new weight parameters.

3.2. Cost Function

In this section, we explain the cost function that can check the degree of similarity between the existing matrix and the newly added matrix.

In order to obtain $\hat{X}$, we divided existing matrix into 3 matrices including left singular matrix, singular value matrix, and right singular matrix (*U*, *S*, *V*) through formula (1).

$$X = USV^T \tag{1}$$

Where *U* is left singular matrix, *S* is singular matrix, and *V* is right singular matrix. After that we induced formula (2) using characteristics, which multiply left and right singular matrix of SVD by transposed matrix in order to obtain identity matrix.

$$S = U^T XV \tag{2}$$

Accordingly, formula (2) is became standard in order to create matrix in which we modified existing SVD to obtain singular value.

The created matrix *S* is a diagonal matrix and it has unique value regarding right and left singular matrix. From this, we can estimate matrix with added new weight matrices and then obtain matrix via matrix. Accordingly, the cost function which shows difference between exiting matrix *X* and crated new matrix $\hat{X}$ obtained by formula (3). At this time, g() is summation of the diagonal elements of matrix.

$$cost = C = \frac{1}{2}(g(S) - g(\hat{S}))^2 \tag{3}$$

3.3. Update Rule

In this section, gradient descent method is used to update the new matrix $\hat{X}$ to reduce the difference between the original weight matrix X and the newly generated matrix $\hat{X}$. Therefore, the update rule is derived using equation (3) and the cost of equation (3) is reduced through repeat of update rule. That means the newly generated matrix $\hat{X}$ is getting closer to the original matrix X. In order to update the direction to reduce the cost, we need to know how the cost changes with the amount of change the newly generated matrix $\hat{X}$. And use of a partial derivative can be expressed as equation (4) in order to show the change of cost according to the change of matrix $\hat{X}$.

$$\frac{\partial C}{\partial \hat{X}} \tag{5}$$

Because we are only concerning ourselves with the $\hat{X}$, so we can substitute the cost. And for convenience of formula derivation, the *g()* function is replaced by the letters S and $\hat{S}$.

$$\frac{\partial C}{\partial \hat{X}} = \frac{\partial\left(\frac{1}{2}(S-\hat{S})\right)^2}{\partial \hat{X}} \tag{6}$$

And equation (7) shows split into two derivatives using chain rule. And to find left derivative, we simply apply the general power rule.

$$= \frac{\partial\left(\frac{1}{2}(S-\hat{S})\right)^2}{\partial \hat{S}} \frac{\partial \hat{S}}{\partial \hat{X}} = -(\mathrm{S}-\hat{S}) \frac{\partial \hat{S}}{\partial \hat{X}} \tag{7}$$

In order to derive the final update expression, $\hat{S}$ must be partially differentiated by $\hat{X}$. $\hat{S}$ is a scalar value according to the sum of the diagonal elements of matrix. Therefore, we can derive the gradient function for equation (5) by differentiating the scalar value with a matrix. Equation (8) represents the gradient function for equation (5).

$$\frac{\partial C}{\partial \hat{X}} = -(\mathrm{S}-\hat{S}) \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \cdots & \frac{\partial y}{\partial x_{m1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{1n}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix} \tag{8}$$

Gradient descent method indicates that the change to $\hat{X}$ should be proportional to the gradient. In order to select the proportional constant alpha and remove the minus sign to move it in the negative direction of the gradient to minimize the error, the final equation is as shown in (9)

$$\Delta \hat{X} = \alpha(S-\hat{S}) \frac{\partial \hat{S}}{\partial \hat{X}} \tag{9}$$

3.4. Procedure of WIPT

Table 1 shows how $\hat{X}$ is initialized using the cost function and update rules.

Table 1. Pseudocode of WIPT Algorithm

```
01  Input :
02      existing weight matrix X
03  Output :
04      extended weight matrix X̂
05
06  [S, V, D] = SVD(X)
07  Random initialization to X̂
08  While ( cost < ε )
09      [Ŝ, V̂, D̂] =SVD(X̂)
10      calculate to cost
11          if (cost < ε )
12              break;
13      gradient descent to ΔX̂
14   Copy X to X̂
```

Lines 1 to 4 represent the input and output of the algorithm. The input of the algorithm is the current weight matrix X and the output is the expansion matrix $\hat{X}$ that has been initialized. Line 6 decompose the existing matrix through SVD. And line 7 initializes the matrix $\hat{X}$ before updating to a random value between 0 and 1. Lines 8 to 13 show the process of reducing the cost of the matrix $\hat{X}$ and updating it. First, $\hat{X}$ is decomposed into SVD, and then the cost of the current matrix X and matrix $\hat{X}$ is calculated cost is less than ε, escape the loop and copy the existing matrix X to the newly matrix $\hat{X}$. In this way, an extended matrix $\hat{X}$ is generated, which is similar to an existing matrix and has existing information.

4. Experiments and Results

For the experiments, we compare with whole-learning and WITP algorithm. Data set which used for learning use MNIST data set of UCI. The MNIST data set [16] include 16 features for recognition of character. The procedure of experiments is as bellows:

a. We extract features regarding A and E from the MNIST data set
b. The DNN learn only 15 features regarding A and E.
c. We add 1 feature into the DNN and then compare with whole-learning and partial-learning.

Parameters of neural network for creating model are same as below

a. Neural networks has 15 input units, 8 hidden units, and 2 output units.
b. Hyperparameters:
   - Learning rate : 0.001, momentum : 0.5, layer : 5.
c. For learning, the neural network use backpropagation and Epoch is set as 500 times.

Table 1. Average Results of Two Methods

	Time	Accuracy
Whole-Learning	23.078	90.725
Partial-Learning	10.785	83.247

Table 1 shows the average time and accuracy of the 100 times model generated for whole-learning and WIPT algorithm. As shown Table 1, the proposed WIPT algorithm has lower accuracy value than existing whole-learning. This is because the proposed WIPT algorithm has updating weight whenever it learns added new features. However, the proposed WIPT algorithm only added new features without learning whole training data. Accordingly, it has lower learning time than whole-learning.

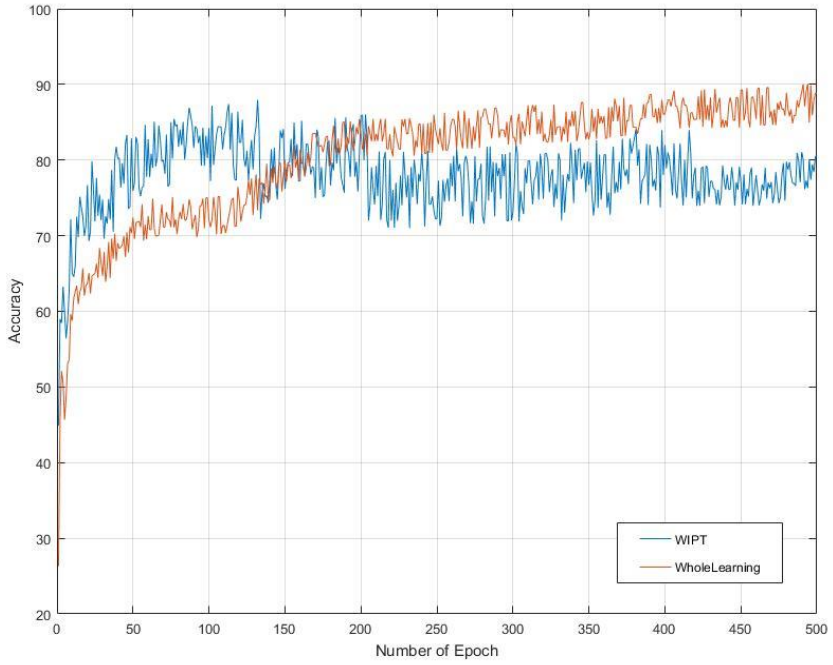

Figure 4. Accuracy Evaluation

Figure 4. shows the change in accuracy with the number of epoch. Typical neural networks shows low accuracy when there are not many epoch times. And as the number of epoch increases, it converges with high accuracy. This experiment is also same. But, in the case of the WIPT algorithm, the accuracy increases rapidly at a low epoch, and then the accuracy decreases at intervals of 150 ~ 200 epoch. It is not shown in the graph, if we epoch about 750 times, we can see that the accuracy is similar to that of whole learning. This is because when the number of epoch is small at the beginning, it is presumed that the weight is largely adjusted because the magnitude of the error with respect to the weight of the newly added feature is large, and when the fine tune of the newly added weight is completed, every weight is fine tune.

5. Conclusion

In this paper, we propose a WIPT algorithm to expand the latent matrix based on existing weight matrices to learn only newly added features and parameters in order to improve the efficiency learning. The proposed algorithm aims to maximize the temporal

efficiency of learning by learning only newly added data as a potential matrix with newly existing information. In this way, data can be learned in a short time than whole learning, which re-learns the whole data. However, there is a problem that the accuracy is lower than the existing learning method. In order to improve to the existing accuracy level, it is necessary to spend learning time similar to the whole learning. Also, it takes some time to decompose the matrix with SVD and update it using the gradient descent method. Therefore, future work is needed to overcome the above problems.

Acknowledgments

The present Research has been conducted by the Research Grant of Kwangwoon University in 2016.

References

[1] Y. Kodratoff and R. Michalski, "Machine Learning: An Artificial Intelligence Approach", vol.3, Morgan Kaufmann Publishers, **(2014)**.
[2] C. Cai, Y. Xu, D. Ke and A. Su, "A Fast Learning Method for Multilayer Perceptrons in Automatic Speech Recognition Systems", Journal of Robotics, Hindawi Publishing, vol. 2015, **(2015)**.
[3] R. O. Duda, P. E. Hart and D. G. Stork, "Pattern Classification – Second Edition", A Wiley-Interscience Prulication, **(2012)**.
[4] Y. LeCun, Y. Bengio and G. E. Hinton, "Deep Learning", Nature, Nature 521, **(2015)**, pp. 436-444.
[5] A. Krizhevsky, B. Sutskever and G. E. Hinton, "ImageNet Classfication with Deep Convolutional Neural Networks", NIPS, **(2012)**.
[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions", Computer Vision and Pattern Recognition, **(2015)**.
[7] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", Computer Vision and Pattern Recognition, **(2015)**.
[8] A. Graves, A. R. Mohamed and G. E. Hinton, "Speech Recognition with Deep Recurrent Neural Networks", Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), **(2013)**.
[9] L. Deng, G. E. Hinton and B. Kingsbury, "New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: an Overview", Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), **(2013)**.
[10] N. Morgan, "Deep and Wide: Multiple Layers in Automatic Speech Recognition", IEEE Transactions on Audio, Speech and Language Processing, vol. 20, no. 1, **(2012)**, pp. 7-13.
[11] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", Journal of Machine Learning Research (JMLR), **(2011)**.
[12] T. Tieleman and G. Hinton, "Lecture 6.5 rmsprop. Technical Report MSU-CSE-00-2", **(2012)**.
[13] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", The 3rd International Conference for Learning Representations", **(2015)**.
[14] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks".
[15] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-level Performance on ImageNet Classification", The IEEE International Conference on Computer Vision (ICCV), **(2015)**.
[16] Letter Recognition Data Set, http://archive.ics.uci.edu/ml/datasets/Letter+Recognition.

