Deep Online Sequential Extreme Learning Machines and Its Application in Pneumonia Detection

Sriram Vijendran
SRM Institute of Science and Technology
e-mail: vijendran.sriram@gmail.com

Rahul Dubey
Robert Bosch Engineering and Buissness Solutions
Private Limited, India
e-mail: RahuKumar.Dubey@in.bosch.com

Abstract—**Deep neural networks have demonstrated high levels of accuracy in the fields of image classification. Deep learning is a multilayer perceptron artificial neural network algorithm, that uses a backpropagation based learning technique to approximate complicated functions and alleviating the difficulty associated with optimizing deep models. Multilayer extreme learning machine (MLELM) is a learning algorithm of an artificial neural network which takes advantages of deep learning and extreme learning machine. Not only does MLELM approximate the complicated function but it also does not need to iterate during the training process. Furthermore, Online Sequential Extreme Learning Machines (OSELM) is an adaptive algorithm based on ELM that does not require fresh training when faced with a new dataset, but can adapt to the new dataset by being trained on the new dataset alone. We combining MLELM and OSELM put forward Multilayer OSELM and apply it to the Pneumonia Chest X-Ray image dataset in this paper. By simulating and analysing the results of the experiments, effectiveness of the application of Multilayer OSELM in Pneumonia identification is confirmed.**

Keywords-Deep Learning (DL); Support Vector Machine (SVM); CNN; Online Sequential Extreme Learning Machine (OSELM)

I. INTRODUCTION

With the boom of computers and the exponential improvements of computations, machine learning has found more applications in everyday life and problems such as security systems, fault detection, social media etc. It has been used to forecast stock prices, identify structural faults in buildings, image classification, images segmentation etc. While classical machine learning algorithms show good performance in pre-processed data, they have been insufficient when dealing with unstructured data such as images, audio files, video files etc. In these cases, deep learning algorithms such as Neural networks have significantly superior performance with the trade-off requiring more data and more time to train along with more processing power [1-5].

Extreme learning machines are Single Layer Feed-forward Neural Networks that have been thoroughly discussed by several researchers for image classification applications. Two main architecture exist for SLFN, namely: (1) those with a single layer and (2) those with an auto encoding layer. For most applications that use SLFN's, batch-learning type methods are implemented. These are often time consuming processes that require proper tuning hyper parameters such as learning rates, number of epochs, stopping criteria etc. Also batch-learning algorithms classify new data using old examples, and hence require retraining. Extreme learning machines are a new type of learning scheme for SLFN's that provide better generalization and faster learning speeds on a number of benchmark problems in and engineering applications in regression and classification areas [6-10].

This paper presents an ELM based comprehensive adaptive image classifier for the detection of pneumonia by the use of Chest X-Rays for faster diagnosis and earlier delivery of treatment. A mathematical model was developed and analysed in Python programming language. The applicability of such an adaptive model is discussed. Deep OS-ELM with sigmoid activation function is used in developing the proposed model. In the following sections we will review the methods of support vector machines, extreme learning machines and its variants, and CNN's [11].

II. THEORY

A. Shallow Models

1) Support vector machines

In this section, the principle of SVM for classification [12-13] problems is briefly reviewed. Given a training set of N data points $\{x_i, y_i\}_{i=1}^{N}$, where the label $y_i \in \{0,1\}$,i=1,...,N. According to the structural risk minimization principle, SVM aims at solving the following risk bound minimization problem with inequality constraint.

$$\frac{1}{2}(\|w\|)^2 + C\sum_{i=1}^{N}\Im_i \qquad (1)$$

where φ(·) is a linear/nonlinear mapping function, w and b are the parameters of classifier hyper plane.

Generally, for optimization, the original problem of SVM can be transformed into its dual formulation with equality constraint by using Lagrange multiplier method. One can construct the Lagrange function $L(\text{w,b},\Im_i,\alpha_i,\lambda_i)$ as.

$$\boldsymbol{\frac{1}{2}(\|w\|)^2 + C\sum_{i=1}^{N}\Im_i - \sum_{i=1}^{N}\alpha_i(y_i[W^T\phi(x_i)+\text{b}] - 1 + \Im_i) - \sum_{i=1}^{N}\lambda_i\Im_i} \qquad (2)$$

where $\alpha_i \geq 0$ and $\lambda_i \geq 0$ are Lagrange multipliers. The solution can be given by the saddle point of Lagrange function by solving,

$$\min_{\alpha_i,\lambda_i}\ \max_{w,b,\Im_i} L(w, b, \Im_i; \alpha_i, \lambda_i) \qquad (3)$$

By calculating the partial derivatives of Lagrange function (2) with respect to w, b and $\Im_i$, one can obtain

$$\frac{\partial \mathbf{L}(w,b,\Im_i,\alpha_i,\lambda_i)}{\partial \mathbf{w}} = \mathbf{0} \rightarrow \mathbf{w} \sum_{i=1}^{N} \alpha_i \mathbf{y_i} \boldsymbol{\phi}(\mathbf{x_i})$$
$$\frac{\partial \mathbf{L}(w,b,\Im_i,\alpha_i,\lambda_i)}{\partial \mathbf{b}} = \mathbf{0} \rightarrow \sum_{i=1}^{N} \alpha_i \mathbf{y_i} = \mathbf{0} \quad (4)$$
$$\frac{\partial \mathbf{L}(w,b,\Im_i,\alpha_i,\lambda_i)}{\partial \Im_i} = \mathbf{0} \rightarrow \mathbf{0} \leq \alpha_i \leq \mathbf{C}$$

Then one can rewrite (3) as

$$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \phi(x_i)^T \phi(x_j) \quad (5)$$

By solving α of the dual problem (5) with a quadratic programming, the goal of SVM is to construct the following decision function (classifier),

$$f(x)=\text{sgn}(\sum_{i=1}^{M} \alpha_i y_i \kappa(x_i,x) + b) \quad (6)$$

where $\kappa(.)$ is a kernel function.

2) Extreme learning machines

ELM has the advantage of fast training speed and high generalization performance, but ELM also has the disadvantage of bad robustness.

ELM aims to solve the output weights of a single layer feed-forward neural network (SLFN) by minimizing the squared loss of predicted errors and the norm of the output weights in both classification and regression problems. We briefly introduce the principle of ELM for classification problems.

Given a dataset $X=[x_1,x_2,....,x_N] \in \Re^{dxN}$ of N samples with labels $T=[t_1,.....,t_N] \in \Re^{cxN}$ where c is the number of classes and d is the dimension of the sample. If x_i(i=1,.....,N)belongs to the k^{th}class, then thek^{th}position of t_i(i=1,.....,N)is set as 1 and 0 otherwise. The hidden layer output matrix **H** with *L* hidden neurons can be computed as

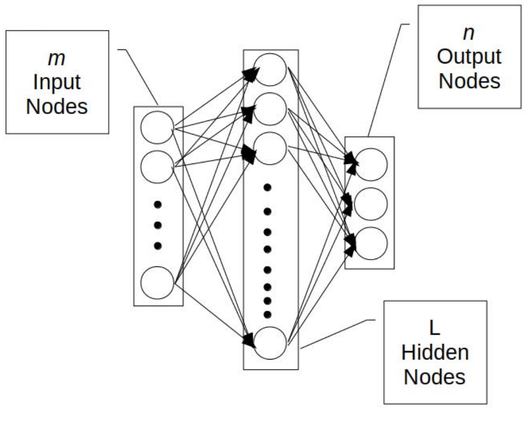

Figure 1. Simple ELM

$$\mathbf{H}=\begin{bmatrix} h(w_1^T x_1+\mathbf{b_1}) & \cdots & h(w_1^T x_1+\mathbf{b_1}) \\ \vdots & \ddots & \vdots \\ h(w_1^T x_1+\mathbf{b_1}) & \cdots & h(w_1^T x_1+\mathbf{b_1}) \end{bmatrix} \quad (7)$$

where h(.) is the activation function of the hidden layer $W=[w1,.......,w_L] \in \Re^{dxL}$ and $B=[b_1,.....,b_L]^T \in \Re^L$ are randomly generated input weights and bias between the input layer and hidden layer. With such a hidden layer output matrix **H**, ELM can be formulated as follows

$$\beta=H^{\dagger}T=(\mathbf{H^T H})^{-1}\mathbf{H^T T} \quad (8)$$

where $\beta \in \Re^{cxL}$ denotes the output weights between hidden layer and output layer, $\xi=[\xi_1,...,\xi_N]$ denotes the prediction error matrix with respect to the training data, and C is a penalty constant on the training errors.

The closed form solution β of (8) can be easily solved. First, if the number N of training patterns is larger than L, the gradient equation is over-determined, and the closed form solution of (8) can be obtained as

$$\beta=H^{\dagger}T=(\mathbf{H^T H}+\frac{\mathbf{I_{LxL}}}{\mathbf{C}})^{-1}\mathbf{H^T T} \quad (9)$$

where I_{LxL} denotes the identity matrix with size of L, and $H^{\dagger}$ is the Moore-Penrose generalized inverse of H. If the number N of training patterns is smaller than L, an under-determined least square problem would be handled. In this case, the solution of (9) can be obtained as

$$\beta=H^{\dagger}T=H^T(\mathbf{H^T H}+\frac{\mathbf{I_{NxN}}}{\mathbf{C}})^{-1}\mathbf{T} \quad (10)$$

Then the predicted output for a new observation **z** can be computed as

$$y=h(z)\beta \quad (11)$$

3) Extreme learning machines-AutoEncoder

Autoencoder is an artificial neural network model which is commonly used in deep learning. Autoencoder is an unsupervised neural network, the outputs of autoencoder are the same as the inputs of autoencoder, and autoencoder is a kind of neural networks which reproduces the input signal as much as possible.

The model of ELM-AE constituted input layer, single-hidden layer, and output layer. The model structure of ELM-AE is shown below, with *d* input layer nodes, *L* hidden layer nodes, *n* output layer nodes, and the hidden layer activation function h(.). According to the output of the hidden layer representing the input signal, ELM-AE can be divided into three different representations as follows.

d > L: Compressed Representation: this represents features from a higher dimensional input signal space to a lower dimensional feature space.

d = L: Equal Dimension Representation: this represents features from an input signal space dimension equal to feature space dimension.

d < L: Sparse Representation: this represents features from a lower dimensional input signal space to a higher dimensional feature space.

There are two differences between ELM-AE and traditional ELM. Firstly, ELM is a supervised neural network and the output of ELM is label, but ELM-AE is an unsupervised neural network and the output of ELM-AE is the same as the input of ELM-AE. Secondly, the input weights of ELM-AE are orthogonal and the bias of hidden layer of ELM-AE is also orthogonal, but ELM is not so.

For N distinct samples, $x_i \in \Re^{dxN}$, (i=1,2,...,N) , the outputs of ELM-AE hidden layer $\bar{X}$ can be expressed as (18), and the numerical relationship between the outputs of the hidden layer and the outputs of the output layer can be expressed as (19):

$$\bar{\mathbf{X}}=\mathrm{h}(ax+b) \quad \mathbf{h}(\mathbf{x_i})\beta=\mathbf{x_i^T} \tag{12}$$

Using ELM-AE to obtain the output weights β can be also divided into three steps, but the calculation method of the output weights β of ELM-AE is different from the calculation method of the output weights β of ELM.

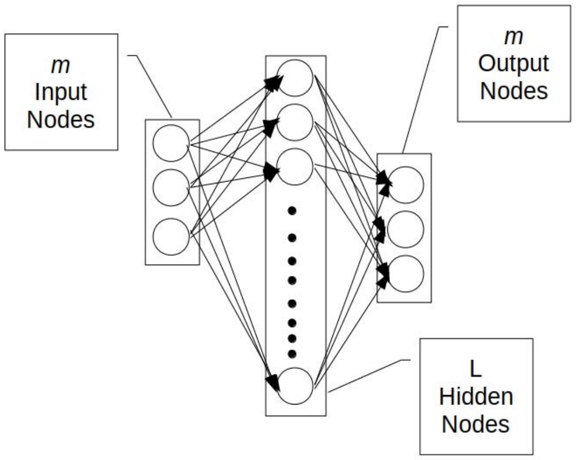

Figure 2. ELM AutoEncoder

B. Deep Models

1) Multilayer extreme learning machine

Multilayer Extreme Learning Machine (MLELM) makes use of ELM-AE to train the parameters in each layer, and MLELM hidden layer activation functions can be either linear or nonlinear piecewise. If the activation function of the MLELM ith hidden layer is h(.), then the parameters between the MLELM ith hidden layer and the MLELM (i − 1) hidden layer (if i − 1 = 0, this layer is the input layer) are trained by ELM-AE, and the activation function should be h(.), too. The numerical relationship between the outputs of MLELM ith hidden layer and the outputs of MLELM (i − 1) hidden layer can be expressed as

$$H_i=\mathrm{h}((\beta_i)H_{i-1}) \tag{13}$$

where H_i represents the outputs of MLELM ith hidden layer (if i−1 = 0, this layer is the input layer, and H_{i-1} represents the inputs of MLELM).

The model of MLELM , β_i represents the output weights of ELM-AE, the input of ELM-AE is H_{i-1} , and the number of ELM-AE hidden layer nodes is identical to the number of MLELM ith hidden nodes when the parameters between the MLELM ith hidden layer and the MLELM (i − 1) hidden layer are trained by ELM-AE. The output of the connections between the last hidden layer and the output layer can be analytically calculated using regularized least squares.

2) Online sequential extreme learning machine

In the derivation of sequential ELM, only the specific matrix H is considered, where H ∈ R N× Ñ , N ≥ Ñ , and rank(H) = Ñ the number of hidden neurons. Under this condition, the following implementation of pseudo inverse of H is easily derived and given by

$$\boldsymbol{H}^{\dagger} = (\boldsymbol{H^T H})^{-1}\boldsymbol{H^T} \tag{14}$$

Which is often called the left pseudo inverse of H from the fact that $H^{\dagger}H = I_{\tilde{N}}$. This equation is called the *least-squares solution* of equation (15). The sequential implementation of equation (15) results in our sequential ELM. Hereafter we talk about the sequential implementation of least-squares solution of equation (15) followed by the derivation of sequential ELM algorithm. The sequential implementation of equation (15) can be derived and referred as the recursive least-squares (RLS) algorithm.

Pseudocode

RLS Algorithm: Given Hβ = T, we can define $H_0 = [h_1, \cdot\cdot\cdot, h_{\tilde{N}}]^T$ which is the sub-matrix comprising of the first Ñ rows of H, where rank(H_0) = rank(H) = Ñ , and $T_0 = [t_1, \cdot\cdot\cdot\, t_{\tilde{N}}]^T$.

Then

Start Training

Step 1:**Initialize**

$$M_0 = (H_0^T H)^{-1}$$

and

$$\beta^{(0)}=\mathrm{M}_0 H_0 T_0$$

While (k <= num_batches):

Step 2: For each h_{k+1} ((Ñ+k+1)th row of H) and t_{k+1} ((Ñ+k+1)th row of T), we can estimate:

$$\mathbf{M_{k+1}}=\mathrm{M}_{(\mathbf{k})} - \frac{\mathbf{M_{(k)}h_{(k+1)}h^T_{(k+1)}M_{(k)}}}{\mathbf{1+h^T_{(k+1)}M_{(k)}h_{(k+1)}}} \tag{15}$$

$$\boldsymbol{\beta}^{(\mathbf{k+1})}=\boldsymbol{\beta}^{(\mathbf{k})}+\mathrm{M}_{\mathbf{k+1}}\mathbf{h_{k+1}}(\mathbf{t^T_{k+1}} - \mathbf{h^T_{k+1}}\boldsymbol{\beta}^{(\mathbf{k})}) \tag{16}$$

Stop training

After training, predictions for a new observation **z** is predicted by

$$\mathbf{h}(\mathbf{z})\beta=\mathrm{y} \tag{17}$$

3) Convolution neural networks

The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map. As a result of convolution in neuronal networks, the image is split into perceptron's, creating local receptive fields and finally compressing the perceptron's in feature maps of size $m_2 \times m_3$.Thus, this map stores the information where the feature occurs in the image and how well it corresponds to the filter. Hence, each filter is trained spatial in regard to the position in the volume it is applied to. In each layer, there is a bank of m^l filters. The number of how many filters are applied in one stage is equivalent to the depth of the volume of output feature maps. Each filter detects a particular feature at every location on the input. The output $Y_i^{(l)}$ of layer l consists of m_1^l feature maps of size $m_2^{(l)} \times m_3^{(l)}$. The i^th feature map, denoted by $Y_i^{(l)}$, is computed as:

$$\mathbf{Y_i^{(l)}} = \sum_{\mathbf{j=1}}^{\mathbf{m_1^{(l-1)}}} \mathbf{K_{i,j}^{(l)}} * \mathbf{Y_j^{(l-1)}} \tag{18}$$

where K is a filter.

The result of staging these convolutional layers in conjunction with the following layers is that the information of the image is classified like in vision. That means that the pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, and objects into scenes. The pooling or down sampling layer is responsible for reducing the spacial size of the activation maps. In general, they are used after multiple stages of other layers in order to reduce the computational requirements progressively through the network as well as minimizing the likelihood of over fitting. There are several types of pooling used, of which we will concentrate on max pooling as it gave a better performance on the test set.

III. Experiment and Analysis

In this section, the performance of Multilayer Online Sequential Extreme Learning Machine is compared with more popular algorithms such as CNN, SVM, and Regularized Extreme Learning Machines on both the MNIST and Chest X-Ray for Pneumonia. There are 60,000 training images and 10,000 testing images for MNIST dataset and 3,516 training images and 2,344 testing images in Chest X-Ray for Pneumonia dataset. All simulation were carried out in Python=3.6 running on a 2GHz PC with a 12GB NVIDIA K80 Tesla GPU. The activation function used in the Deep OS-ELM s a simple sigmoidal function g(x) = 1/(1 + exp(−x)). 30 trials were conducted for Deep OS-ELM.As observed from Table I, OS-ELM can obtain better generalization performance than other popular learning algorithms in these real cases and complete learning at very fast speed.

The performance comparison of Multilayer OS-ELM with SVM, RELM, and CNN on MNIST datasets is shown in Table I. It is clearly observed that Multilayer OS-ELM testing accuracy is higher than RELM and SVM, either the average or the maximum, and the best values of Multilayer OS-ELM testing accuracy are higher than RELM and SVM, and best testing and training times of Multilayer OS-ELM is better than CNN.

As shown in Figure 3, we can make choices that the numbers of RELM hidden layer nodes on MNIST dataset are and while the regularized parameter C for RELM is 0.5. In case of the OS- ELM, the number of hidden nodes are. The Multilayer ELM and Multilayer OS-ELM have two hidden layers. The first layers behave as auto-encoding layers and the final layers as classification layers. In both graphs, the blue line represents the training accuracy, while the red represents the testing accuracy. Figure 4 shows the selection of number of hidden nodes in the autoencoding layer in Multilayer OS-ELM and Figure 5 shows the selection of number of hidden nodes of classification layer in Multilayer OS-ELM on the MNIST dataset.

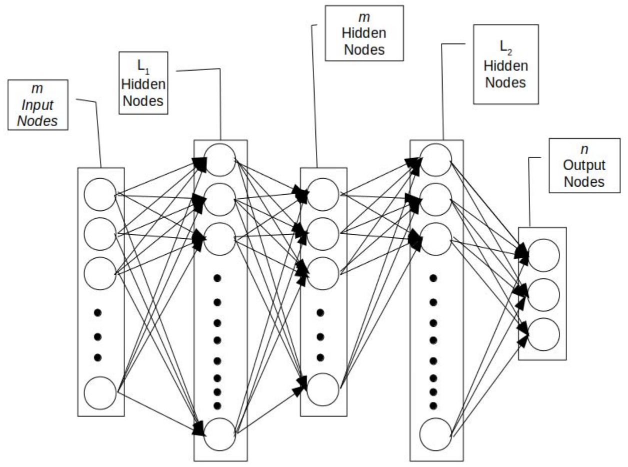

Figure 3. Multilayer ELM

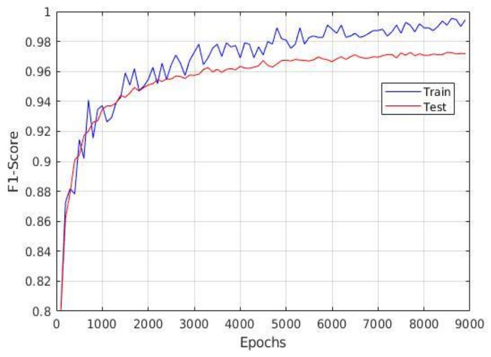

Figure 4. MNIST Output Performance

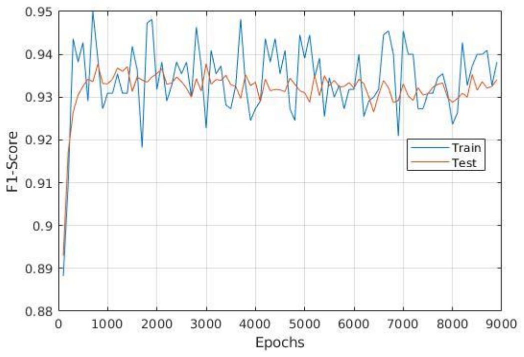

Figure 5. MNIST AutoEncoder Performance

Figure 6 shows the selection of number of hidden nodes in the autoencoding layer in Multilayer OS-ELM and Figure 8 shows the selection of number of hidden nodes of classification layer in Multilayer OS-ELM on the Chest X-Ray for Pneumonia. As depicted by the Fig. 6 and 7, the best configuration of the OS-ELM is Layer 1 with 2300 nodes and Layer 2 with 1400 nodes. This configuration achieves 92.5% accuracy on the test dataset.

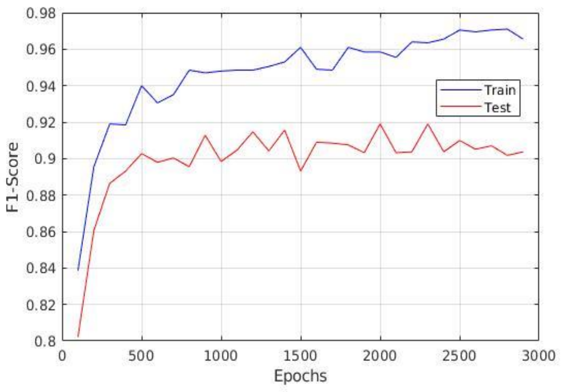

Figure 6. Pneumonia AutoEncoder Performance

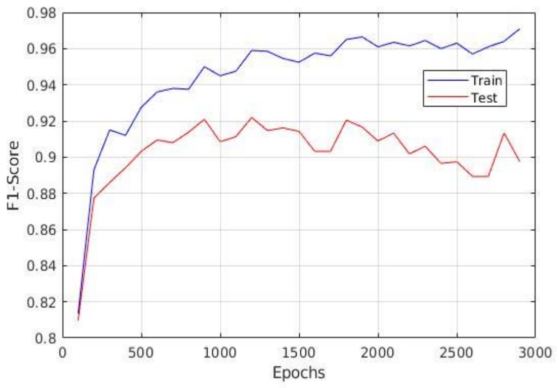

Figure 7. Chest Output Performance

TABLE I. MODEL PERFORMANCE ON MNIST DATASET

Model	Data	Accuracy	Training Time	Testing Time
SVM	Train Test	95.8% 93.5%	3687.2s	1.2s
CNN	Train Test	98.1% 97%	179.58s	1.034s
ELM	Train Test	94.28% 92.08%	31.71s	0.098s
Multilayer ELM	Train Test	93.1% 91.1%	72.71s	0.15s
OS-ELM	Train Test	98% 96.95	168.678s	0.93s
Multilayer OS-ELM	Train Test	98.7% 97%	202.0401s	0.234s

TABLE II. MODEL PERFORMANCE ON CHEST X-RAY DATASET

Model	Data	Accuracy	Training Time	Testing Time
SVM	Train Test	87.8% 86.5%	201.76s	2.01s
CNN	Train Test	94.1% 92%	1980.74s	2.434s
ELM	Train Test	90.2% 89.8%	106.71s	0.158s
Multilayer ELM	Train Test	91.1% 90.1%	117.71s	0.28s
OS-ELM	Train Test	95% 90.95	130.678s	1.03s
Multilayer OS-ELM	Train Test	96% 91.7%	150.0401s	1.23s

IV. CONCLUSION

This paper presents the potential application of Deep OS-ELM's in the field of medical imaging studies, specifically in the detection of pneumonia from Chest X-Rays. Detailed modelling and analysis of images are presented. The proposed model has shown better performance than SVM's and ELM's in terms of accuracy and better training time than CNN's. Test results indicate that the proposed model is highly effective in the diagnosis of pneumonia through Chest X-Rays and can be implemented easily with minimal computational requirements. Future work will make use of a more wide variety of kernels and more number of hidden layers.

REFERENCES

[1] Shifei Ding, Nan Zhang, Xinzheng Xu, Lili Guo, and Jian Zhang Deep Extreme Learning Machine and Its Application in EEG Classification

[2] Shifei Ding, Xinzheng Xu, Nie ru, Extreme learning machine: algorithm, theory and applications

[3] Guang-Bin Huang, Nan-Ying Liang, Hai-Jun Rong, P. Saratchandran, and N. Sundararajan, On-Line Sequential Extreme Learning Machine, the IASTED International Conference on Computational Intelligence (CI 2005), Calgary, Canada, July 4-6, 2005

[4] Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol 2, no 25–29, pp 985–990

[5] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[6] G.-B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, Int. J.Mach. Learn. Cybern. 2 (2011) 107–122

[7] .E. Cambria, G.-B. Huang, Extreme learning machines, IEEE Intell. Syst. 28 (2013) 30–31.

[8] G.-B. Huang, An insight into extreme learning machines: random neurons, random features and kernels, Cognit. Comput. 6 (2014) 376–390.

[9] A. Basu, S. Shuo, H. Zhou, M.H. Lim, G.-B. Huang, Silicon spiking neurons for hardware implementation of extreme learning machines, Neurocomputing 102 (2013) 125–134.

[10] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (2006) 1411–1423.

[11] Rawat, Waseem & Wang, Zenghui. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Computation. 29. 1-98. 10.1162/NECO_a_00990.

[12] Evgeniou, Theodoros & Pontil, Massimiliano. (2001). Support Vector Machines: Theory and Applications. 2049. 249-257. 10.1007/3-540-44673-7_12.

[13] Xiaowu Sun, Lizhen Liu, Hanshi Wang, Wei Song and Jingli Lu, "Image classification via support vector machine," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Harbin, 2015, pp. 485-489.doi: 10.1109/ICCSNT.2015.7490795