Comparison of optimizers' performance in the compression of MNIST handwriting

Prabal Devkota
BE graduate
Kathmandu University
Dhulikhel, Nepal
prabaldevkota19@gmail.com

Ram Kaji Budhathoki*
Associate Professor
Kathmandu University
Dhulikhel, Nepal
ram.budhathoki@ku.edu.np

Abstract: An undercomplete autoencoder model learns the principal features from the input Modified National Institute of Standards and Technology (MNIST) training datasets. The reliability of principal features in codeword depends on loss convergence ability of optimizer and activation function. In this paper, the convergence ability of optimizers and activation functions are studied by training the model for 100 epochs. The result shows that Adaptive Moment Estimation (Adam) optimizer exhibits sharp decay of Mean Square Error (MSE) both in linear and non-linear activation functions. The non-linear activation outperforms the linear activation during compression of MNIST handwriting. This result would be helpful in dimensionality reduction application such as image compression.

Keywords: Undercomplete autoencoder, Optimizer performance, compression, MNIST, activation function

I. INTRODUCTION

The term 'auto association' was used on Multilayer Perceptron Network (MLPN) used for dimensionality reduction application like image compression working on auto-association mode. The non-linearities unit of MLPN are useless and optimal parameter values can be derived directly by pure linear techniques relying on Single Value Decomposition (SVD) [1].

An auto association exhibited by linear activation is a special case of Principal Component Analysis (PCA). The error function E has a unique local and global minimum while others are saddle points [2].

The development of deep belief network, layer wise pretraining of deep network, dimensionality reduction and stacking of Restricted Boltzman Machines (RBMS) are major paradigm leading to the development of deep learning autoencoder. The effective way of initializing weight allows autoencoder to learn low-dimensional code better than PCA [3].

In unsupervised learning, pretraining of Artificial Neural Network (ANN) subjects the neural net towards minima, minimizing variance and introducing bias. The effect of pretraining in unsupervised learning has similar effect of regularization in neural network [4].

The motive of this research work is to find the right optimizer and activation function that exhibits early convergence of loss function. The basis for assumption is that linear activation in the hidden layer should exhibit or outperform non-linear activation as dimensionality reduction is a special case of linear PCA inferred from [1],[2]. The unimodal error surface in the linear model should reach towards global minima early; thereby, helping the model to learn feature in a reduced dimension quickly and more appropriately. The non-linear activation function and optimizer with an ability to learn feature would be useful to pretrain supervised model exhibiting non-linearities.

II. FRAMEWORK AND METHODS

A. *Under complete autoencoder*

An undercomplete autoencoder shown in the Fig. 1 consists of a smaller number of neurons in a hidden layer than the input layer. The codeword h passing through various encoding layers f can be written as $h = f(X)$, and the output Y passing through decoding layers g can be written as $Y = g(h) = g(f(X))$. The learning in under complete auto-encoder is achieved without prior information or label, therefore undercomplete autoencoder follows unsupervised learning approach. The cost function is defined such that architecture is forced to replicate input neurons in output layer and can be expressed mathematically by the MSE loss function $J(W, b; x, y)$ in equation (1) [5].

$$J(W,b;x,y) = \frac{1}{m}\sum_{i=1}^{m}\left(0.5 * (y_{(i)} - x_{(i)})^2\right) \qquad (1)$$

Where, m is the batch size used in training the model, x and y are the input and output neuron vectors respectively. The biases and weights of the model gets updated by the gradient descent algorithm and their adjustment occurs by backpropagation [5]. The mini-batch gradient descent convergence can be accelerated by using optimizers. The optimizers used in the model are listed below.

1. Adaptive Gradient (AdaGrad)
2. Root Mean Square Propagation (RMSProp)
3. Adadelta
4. Adaptive Moment estimation (Adam)

Also, the activations sigmoid, linear and Rectified Linear Unit (ReLU) are used in training the model. All of these activation functions are non-linear except linear activation.

B. *Experiment dataset and methodology*

The MNIST data set [6], shown in Fig. 2, consists of 60,000 training set examples and 10,000 test set examples. The data set required for training is retrieved using Keras Application Programming Interface (API) [7]. The gray scale image of 28*28 image centered towards the center of mass of pixel is normalized first and reshaped to arrays of size 784.

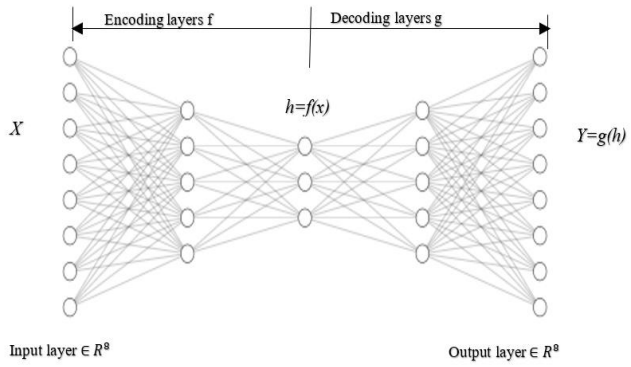


Fig. 1: Undercomplete autoencoder model used in image compression

A seven layers autoencoder neural network architecture is created in Python using Keras API. The first layer consists of 784 input neurons, and subsequent second and third layers consist of 128 and 64 neurons respectively. The fourth layer, codeword representing layer, consists of 36 neurons. This results in a compression ratio around 22. A decoding layer, implemented as mirror of encoding layer, consists of 64, 128 and 784 neurons respectively.

The weight initialization by using Xavier-Glorot technique and bias initialization using zeros are chosen for the model. The gradient descent algorithm is executed in the batch of size 500 for 100 epochs using different optimizers. The results of different optimizers are obtained when ReLU and sigmoid are respectively used in hidden and output layer (NL-NL activation). Similarly, results are obtained for linear activation in hidden layer and sigmoid in output layer (L-NL activation). This was followed by linear activation both in hidden and output layer (L-L activation).

III. Data Analysis and Results

A. *Data observation of NL-NL activation*

The graph in Fig. 3 (a) shows the loss convergence of different optimizers in NL-NL activation. The mini-batch gradient descent and Adagrad optimizer with learning rate 1 shows quick decay of loss function. The error falls exponentially in RMSprop and Adadelta optimizer with learning rates 0.001 and 1 respectively. The comparative graphs in Fig. 3 (a) show that adam optimizer have a sharp decay curve and the least MSE at the end of 100 epochs. The fading errors curve of Adadelta and Adagrad appears to be overlapping. The similar observation holds true in case of RMSprop and mini-batch gradient descent.

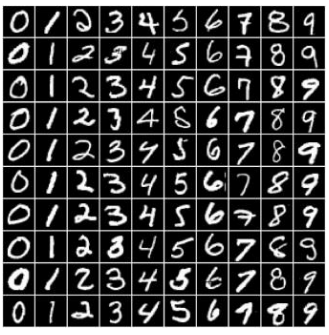


Fig. 2: Samples of MNIST dataset

B. *Data observation of L-NL activation*

In Adam, the learning rate 0.001 exhibits abrupt and earlier decay of loss function outperforming all other optimizers. This is shown in the graph of Fig. 3 (b) The decay curve of Adagrad of learning rate 1 and RMSprop of learning rate 0.001 appears to overlapping. However, observing data up to hundred thousandth decimal point RMSprop outperforms Adagrad at the end of 100 epochs. The similar graph exists in mini-batch gradient descent and Adadelta where Adadelta outperforms at end of 100 epochs.

C. *Data observation of L-L activation*

The graph in Fig. 3 (c) shows error curve of Adam optimizer plunging with least MSE at the end of 100 epochs. The error curve of mini-batch gradient descent and Adadelta optimizers fades following the same trajectory and appears to be overlapping. However, mini-batch gradient outperforms at the end of 100 epochs. During the training, Adagrad optimizer with learning rate 1 gives rise to model instability. Therefore, the learning rate 0.01 is preferred during training. The RMSprop and Adam optimizers exhibits better error decay in L-L activation.

D. *Result and Analysis*

The loss function is given in equation (1). The most of optimization techniques exhibit no significant decay of loss function after 50 epochs. Therefore, MSE at the end of 50 epoch are compared. The graph in Fig. 3 and parameters in Table I are evaluated to infer suitable optimizer and activation function.

The linear model exhibiting unimodal error surface is expected to reach towards global minima quickly; however, such observation is not realized. This might be due to complex features existing within the training dataset. The non-linear activation consisting of multiple local valleys in error surface realizes the feature existing in MNIST dataset more accurately. The experiment results in Table I and Fig. 4 demonstrate that linear activation function in hidden layer along with non-linear activation in the output should be preferred for early convergence of loss function in the compression of MNIST handwriting. The L-NL activation learns complex feature from the dataset and maps the encoded codewords appropriately outperforming both the NL-NL activation and L-L activation.

The non-linear activation with Adam optimizer demonstrates sharp decay of loss function. The concept of first and second orders estimation in Adam optimizer accelerate and prevent annealing of learning rate properly to converge the loss function.

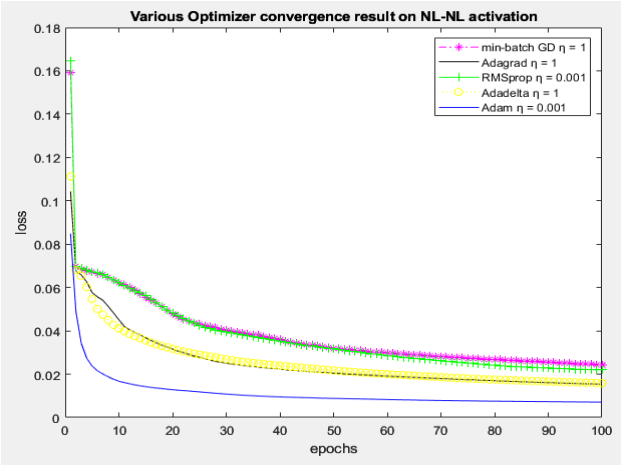
The dimensionality reduction transforms the data from higher input dimensional space to lower dimensional space where variance of the data is maximized. In this simulation, 784 input neurons are represented by 36 neurons in reduced dimension. The observation shows that non-linear activation with Adam optimizer realizes the feature duly and rapidly. Therefore, Adam optimizer with non-linear activation can be used in

dimensionality reduction application such as compression.

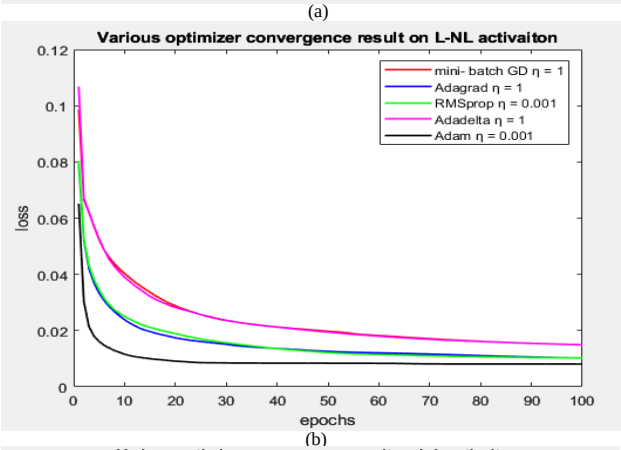
IV. Conclusion

In this paper, convergence of different optimizers and activation functions are analyzed for the MNIST datasets. The loss functions of various optimizers are plotted for linear-nonlinear, linear-linear and nonlinear-nonlinear activation functions at hidden and output layers respectively. The mini-batch gradient descent with Adam optimizer activated by nonlinear function outperforms all other optimizers in the compression of MNIST handwriting.

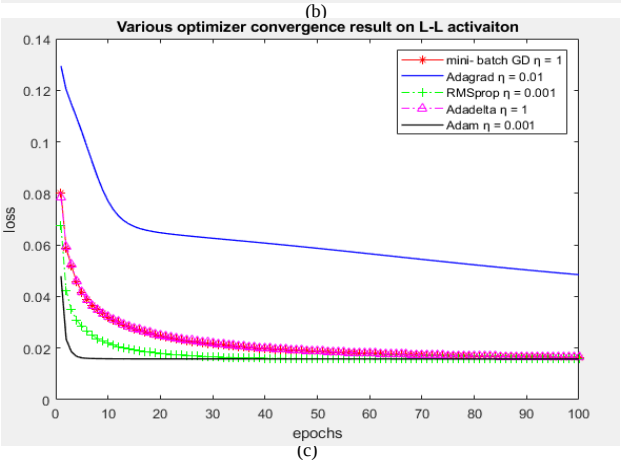
The experiment is performed using MNIST data as a training and test dataset. The inference will be strong if different input dataset results are analyzed.



(a)



(b)



(c)

Fig. 3: Various optimizers epoch Vs loss graph. (a) NL-NL activation. (b) L-NL activation. (c) L-L activation (η is the learning rate)

TABLE I GRADIENT DESCENT (GD) OPTIMIZER MSE LOSS AT END OF 50 EPOCHS

GD optimizer	MSE		
	L-NL Activation	NL-NL Activation	L-L Activation
Mini-batch GD	1.975e-02	3.193e-02	1.856e-02
Adagrad	1.261e-02	2.048e-02	5.871e-02
RMSprop	1.215e-02	2.170e-02	1.591e-02
Adadelta	1.948e-02	3.165e-02	1.877e-02
Adam	8.362e-03	8.802e-03	1.578e-02

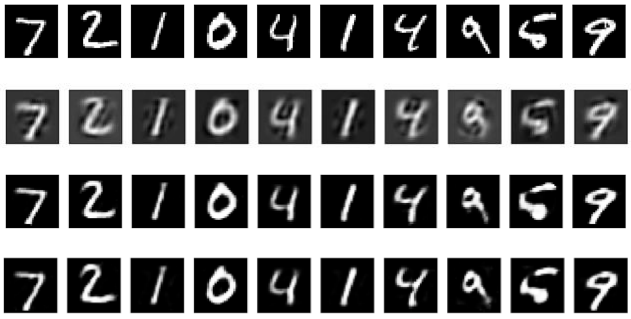


Fig. 4: Original image and reconstruction of predicted image using Adam optimizer at the end of 50 epochs. First row: Original image, second row: L-L activation, third row: NL-NL activation , fourth row: L-NL activation

References

[1] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol Cybern*, vol. 59, no. 4–5, pp. 291–294, Sep. 1988, doi: 10.1007/BF00332918.

[2] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, Jan. 1989, doi: 10.1016/0893-6080(89)90014-2.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science (1979)*, vol. 313, no. 5786, pp. 504–507, Jul. 2006, doi: 10.1126/science.1129198.

[4] D. Erhan, Y. Bengio, A. Courville, P.-A. M. Ca, P. V. Ca, and B. Com, "Why Does Unsupervised Pre-training Help Deep Learning?," 2010. https://dl.acm.org/doi/pdf/10.5555/1756006.1756025 (accessed Mar. 22, 2023).

[5] A. Ng, "Sparse Autoencoder, CS294A Lecture Notes." https://docslib.org/doc/11222999/sparse-autoencoder-cs294a-lecture-notes (accessed Dec. 25, 2022).

[6] Y. LeCun, "MNIST handwritting dataset." https://yann.lecun.com/exdb/mnist/ (accessed Mar. 22, 2023).

[7] "MNIST digits classification dataset." https://keras.io/api/datasets/mnist/ (accessed Mar. 22, 2023).