Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)

Aditya Prasad Mishra
School of Computing Informatics and Decision System Engineering
Arizona State University
Tempe, Arizona 85281
Email: amishr28@asu.edu

***Abstract*—Ongoing research in activation functions, responsible for activating neurons in Multilayer Neural Networks has been producing some exciting results, since the last couple of years. Research in this area is making significant strides towards improving accuracy over large Multilayer Networks.Discovery of ELU as an activation function is one of many such milestones. Initial objective in activation function research was to avoid vanishing gradient effect of a Sigmoid or a Tanh in a MultiLayer Network. Introduction of RELU in neural networks [4] did help us in mitigating the Vanishing gradient effect for both linearly and non linearly separable data. But RELU based units output zero, upon being given a negative input and eventually blocks back propagation. Due to this feature of ReLU the mean activation of incoming nodes shifts away from the origin. Leaky ReLUs [5] and RRelus [6] mitigate this effect of Bias-Shift by ReLU as they by produce a small output in negative area.Both LRELU and RRELU are not very noise robust networks. [1] introduces ELU, a noise robust activation function which not only alleviates the vanishing gradient but also decreases the forward propagated variation and information. Through the course of this project, we tried to prove that ELU is one of the best activation function to use while learning from datasets which requires more than 3 layers for training.The primary test objective was to validate the fact that ELU speeds up training and increases the test accuracy over a multilayer network as suggested by [1]. For our experiments, we would consider a Sigmoid, ReLU and ELU based networks built with the same structural architecture. We will train and test them with the MNIST [7] dataset to analyze the performance of these networks under various constraints. Finally, we will prove that ELU is indeed better than both ReLU and Sigmoid in most of the cases with high dimensional data.**

1. Introduction

Multilayer Neural Network implements linear discriminants with nonlinearly mapped inputs [3]. These networks try to overcome the restrictions faced by 2 and 3 layer networks (with 1 hidden layer) because they have much more expressive power. As per Kolomogorov [3], any continuous valued function $g(x)$ can be represented by (1).

$$\sum_{j=1}^{2n+1} \theta_j (\sum_{i=1}^{d} \sigma_{ij}(x_i)) \tag{1}$$

Equation (1) can be formulated as a Multilayer Neural Network where each of 2n+1 hidden units takes d nonlinear functions as input. So essentially this proves that any continuous valued function can be represented by a Multilayer Neural Network. Because of such high expressive power, Multilayer Neural Networks are utilized as discriminative classifiers to classify objects with a large number of features.In recent years with the advancement of GPU's, their usage is growing in areas like Image Classification, Pattern Recognition, Robotics and Planning in AI.

Use of Multilayer Neural Networks has grown exponentially. Such a growth can be attributed to advancements in Computer Architecture. This was not the case always though. Just about a few years ago, most of the Neural Networks were either two-layered or three-layered. Most of these Networks would have a direct map from input to output with bias and weights. These Networks used Sigmoid or tanh as the activation function. tanh was better than sigmoid as it would take mean activations of incoming nodes closer to zero. But both of the types of units had a cut off for their gradients.

1.1. Vanishing Gradient

When Networks were extended beyond two layers it was found that Sigmoid and tanh due to their limiting Gradients transferred very small gradients to subsequent layers. This lead to a higher training time for weight based learning.It was also observed that gradients were becoming small exponentially which further reduced overall training accuracy of the systems. ReLU was introduced to mitigate this effect. ReLU successfully reduced this effect by producing larger gradients for in all of the hidden layers.

1.2. Bias-Shift Effect

ReLU eliminated Vanishing Gradients but it shifted the mean of activations much far away from zero. If such units do not cancel out each other, they would run the risk of transferring their bias to subsequent layers. This is commonly known as Bias-shift Effect.It is quite evident that this effect increases training time and produces a less accurate classifier overall. LRELU and RRELU are used to deal with this effect. In the current project we will investigate a new activation function ELU and see if it performs better than Sigmoid and ReLU in both training and testing

1.3. Why ELU?

One of the primary objectives of this investigation is to establish a function which decreases the bias-shift effect and pushes the mean activation towards zero (not towards exact zero). It is evident that ELU does possess these qualities. We can easily check this out by analyzing the natural gradient of a Multilayer Network. In theory, it is efficient and does reduce the bias-shift effect. Through live observation, it would satisfy it's acceptance criteria. Hence, Experiments would be designed to find whether ELU can really speed up training in a higher dimensional image classification task by decreasing the bias-shift of nodes. The reason for selecting ELU over LRELU and RRELU is due to its noise robustness. Accuracy comparison between ELU, LRELU and RRELU lies outside of the scope of the current project.

2. Mathematical Formulation

The exponential linear unit (ELU) with $0 < \alpha$ is

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(\exp x - 1) & x \leq 0 \end{cases} \tag{2}$$

$$f'(x) = \begin{cases} 1 & x > 0 \\ f(x) + \alpha & x \leq 0 \end{cases} \tag{3}$$

As per [1] hyperparameter α controls ELUs saturation in negative net inputs. As discussed earlier ELU's does not produce any vanishing gradient effect. Also the Mean activations of ELU is closer to zero. Due to this the gradient is closer to natural gradient which ultimately improves accuracy in training.

2.1. Decrease in Bias-Shift in ELU

Mathematical Analysis of ELU enables us to understand why there is a reduction in Bias-Shift. The bias shift (mean shift) of any unit as per [1] is the change of units mean value due to the weight update. if mean is far from zero such an effect can lead to a decrease in learning.[lecunetall] demonstrates this effect. One of the ideas from [1] use to use a unit natural gradient. Equivalently, we can shift the mean activation towards zero and we will observe the same phenomena of a decrease in bias shift.To achieve this we need a cutting factor. And this is what ELU does. For positive values, it still maintains an identity like ReLU but for negative values, it outputs a value to center the incoming activations.

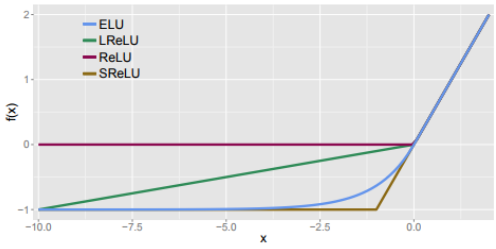

Figure 1. Comparison of Various Activation Functions in RELU Family (Pic Credit - [1])

3. Experiments

To analyze the decrease in Bias-shift by using ELU as mentioned in section 1 and 2, a range of experiments were performed on a series of diverse Multilayer Networks. The objective here was not only to validate this effect but also at the same time visualize the effect by changing in hyper-parameters like a number of nodes or number of layers of the network. The Code used for the design of such experiments is written in Python 3.5.4 and TensorFlow version 1.3.4. Graph plotting was done using Matplotlib library. Internal Tensorflow settings were used for preprocessing MNIST dataset.

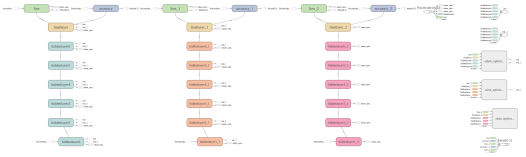

Figure 2. Tensorflow Graph

3.1. Experimental Setup

For our experiments, we used various different types of Multilayer Networks. A total of 6 experiments were performed over three different networks which were similar in structure and differed only in the activation function they use. It was necessary to build these networks in such ways so that a fair comparison can be made in terms of their accuracy and training speedup under various Experimental constraints.

3.1.1. Input and Output Layers. The input layer of our network is a 1-D matrix with 784 values. Each of the nodes corresponds to a pixel value of an MNIST letter. The output layer is again a 1-D matrix with 10 nodes or values. Each of these nodes represents a digit of MNIST.

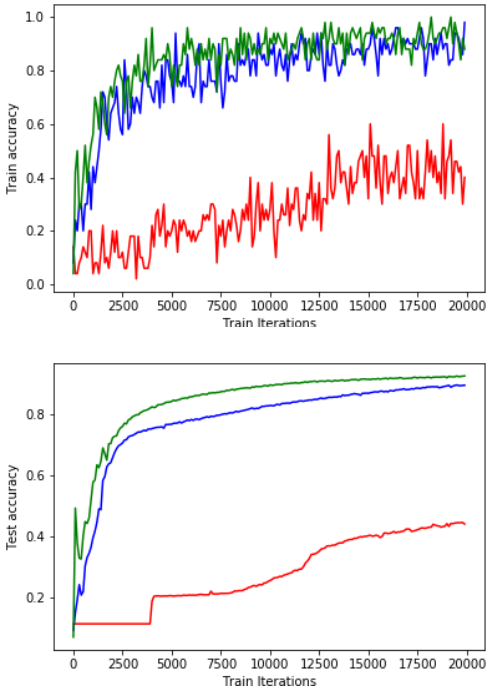

Figure 3. Experiment 1 - 10 layers with 1-Input, 1-Output,1-Softmax and 7-Hidden layers with 16 nodes. ELU clearly outperforms here both during training and during testing.

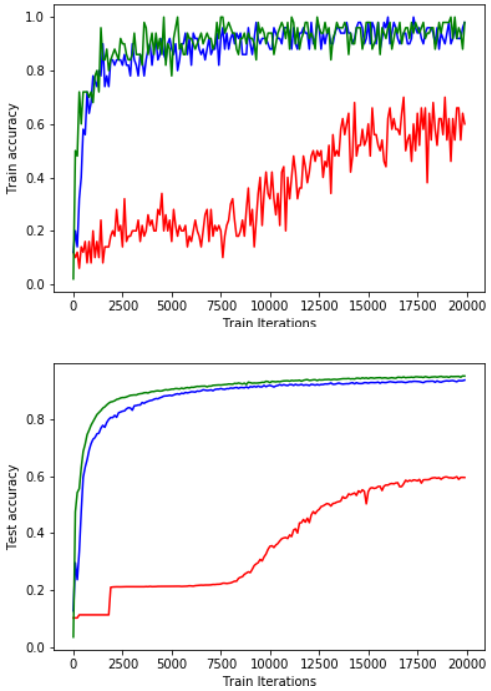

Figure 4. Experiment 2 - 10 layers with 1-Input, 1-Output,1-Softmax and 7-Hidden layers with 32 nodes. ELU clearly outperforms here both during training and during testing. ReLU's performance improves. Sigmoid's performance worsens for most part of the training

3.1.2. Hidden Layers. These layers are matrices of 2-D shape. The Matrix dimensions are always $[NumberofInputNodes * NumberofOutputNodes]$. The first hidden layer in this series have 784 nodes as input and the last hidden layer outputs 10 for MNIST dataset.Rest of the layers stack up in between these two layers and can have a range of input and output nodes depending on the input and output settings of their immediate parent and child.

3.1.3. Loss Function. For our Networks, we use a Softmax layer after the last hidden layer to calculate the Softmax probabilities. Softmax probabilities are relative probabilities of a singular node given a vector of nodes. They are calculated by the formula as $\frac{\exp a_k}{\sum_j \exp a_j}$. After calculating these probabilities the loss function calculates the cross-entropy loss. Cross-Entropy is a concept from information theory and is a probabilistic measure of entropy. It is found out by comparing a true distribution with the estimated distribution. In a mathematical form if p and q are actual and estimated distribution. cross Entropy is $-\sum_x p(x) \log q(x)$. The objective during training is to is to minimize the cross-entropy loss with p being the actual label in the form of an array of 10 nodes and q being the Softmax probabilities.

3.1.4. Optimization and Descent. As described in section 1, the appropriate way to decrease the loss is to find gradients at each node and move towards minima by subtracting the gradients from existing weights. To implement this we used the Adam Optimizer [8] our experiments. Adam is short for adaptive moment estimation. It is computationally efficient then SGD Optimizer and requires less memory. the biggest advantage of using Adam is its adaptiveness. This algorithm adapts the learning rate and sets rate per-parameter resulting in faster training. To avoid any effect in our objective of a fair analysis we made the optimizer same across networks.

3.1.5. Accuracy. We calculate the accuracy by maximizing the output vector and comparing it to the maximum value of the label vector. And using this measure over a batch we calculate the accuracy ratio.

3.1.6. Weight, Bias and Activation. As described in section 1, a Neural Network node is an activation over a discriminant. The activation function we used for testing are Sigmoid, ReLU and ELU. Sigmoid's training is hampered due to Vanishing Gradient.ReLU mitigates the Vanishing gradient but suffers from Bias-Shift.With ELU we must have a speed up in training due to the small output it produces for negative gradient values. As results in section shows that ELU both speeds up the training and provide higher test accuracy. For Initial weight, we use the normal distribution with standard deviation being the $\frac{1}{\sqrt{NoOfNodes}}$ [9]. Variables are initialized only once before the training starts.

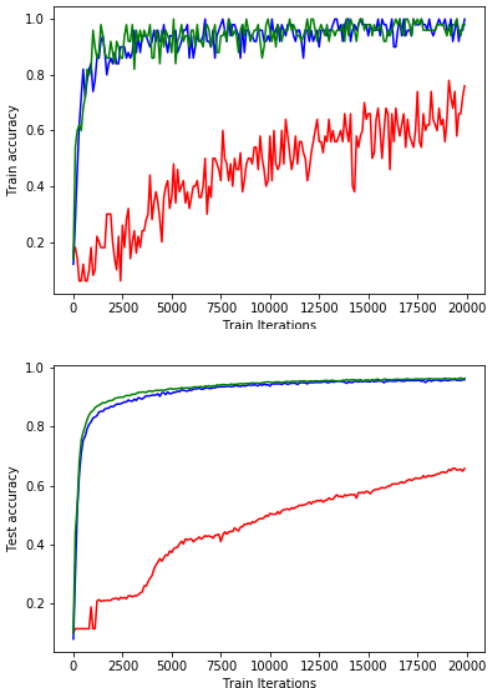

Figure 5. Experiment 3 - 10 layers with 1-Input, 1-Output,1-Softmax and 7-Hidden layers with 64 nodes. ELU performs better during training.Testing performance is almost same with ReLU. Relu shows considerable improvement but Sigmoid stays the same.

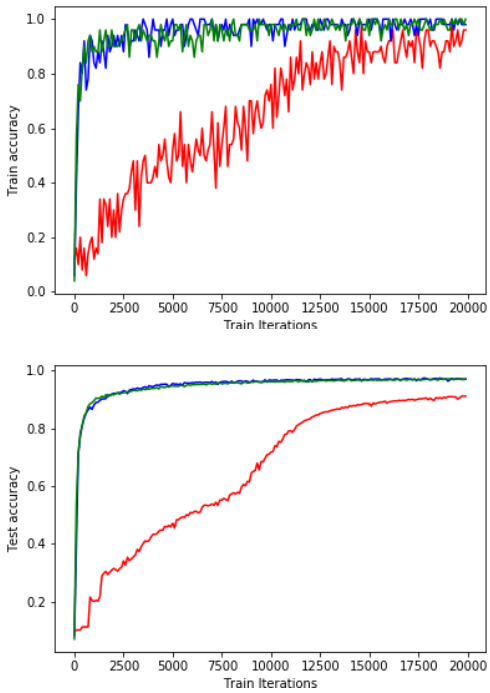

Figure 6. Experiment 4 - 10 layers with 1-Input, 1-Output,1-Softmax and 7-Hidden layers with 128 nodes. During training and testing ReLU and ELU performed the same. Improvement in performance of Sigmoid.

3.1.7. Training. For Training, we used the Adam Optimizer discussed above. Batches of 50 were considered for training in a single iteration. Both training and test accuracy were measured after completion of an Epoch. The objective of the training was to reduce the cross-entropy loss. The training of a single Network consisted of 200 Epochs. Each Epoch had 100 iterations. Accuracy results were stored for each epoch in python arrays and Matplotlib was used to plot the resulting graphs.

3.1.8. Dataset. Dataset used was the MNIST dataset [7]. We used Tensorflow's internal settings were used to download and pre-process the data [9].MNIST is a simple computer vision dataset [9]. It consists of images of handwritten digits like as shown in in figure 9. Each of the images is a 28*28 matrix of pixels. The output vector is a 10 node vector. It has a high value for the correct digit index and rest are zeros. Trained Neural Network work will also generate a similar vector. Both of these vectors are index wised compared using argmax to measure accuracy.

3.2. Experimental Evaluation

3.2.1. Experiment 1-4.

Design - For experiments from 1 through 4, we used a Network structure similar to shown in Figure 1. The designed Network consists of 10 layers with 1-Input, 1-Output,1-Softmax and 7-Hidden Layers. Through experiment 1-4, the parameter that was changed was the number of nodes in each Hidden layer. In experiment 1 We considered 16 nodes. With each subsequent experiment, we doubled the number and the final experiment in this series i.e. 4th experiment consisted of 128 nodes.

Inference - From the graphs output presented from Figure 3-7, it is visualized that ELU did train faster of all of three networks.The classification accuracy improved from Sigmoid to ReLU to ELU.We were also able to visualize the vanishing gradient.Also, it was observed that as we increased the number of nodes of the network, the training performance of ELU and ReLU were almost similar. but with networks with less number of nodes or higher density, it is observed that ELU performed better of the two. In terms of Test Accuracy ELU performed the best in all scenarios.

3.2.2. Experiment 5.

Design - For experiment 5 we used a Network structure similar to shown in figure 2 but with 5 hidden layers instead of 7. The designed Network consists of 10 layers with 1-Input, 1-Output,1-Softmax and 5-Hidden Layers. A number of hidden units were kept at 64 in each layer. Now hen analyzes the results from Experiment 3(figure 5) and Experiment 5(figure 9) as both contains 64 hidden units with 3 containing more hidden layers, We see an interesting result. Both training and test accuracy of ELU remained same. But Test and training accuracies of Sigmoid and Relu

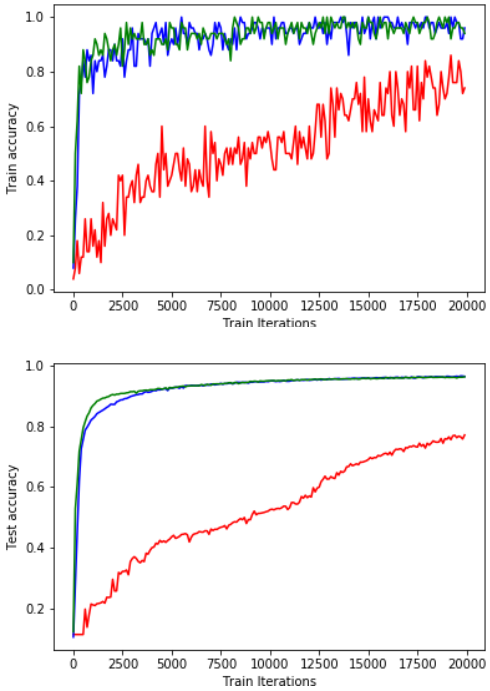

Figure 7. Experiment 5 - 8 layers with 1-Input, 1-Output,1-Softmax and 5-Hidden layers with 64 nodes. Performance ReLU and Sigmoid are much better here than in Experiment 3 with same number of nodes at each layer

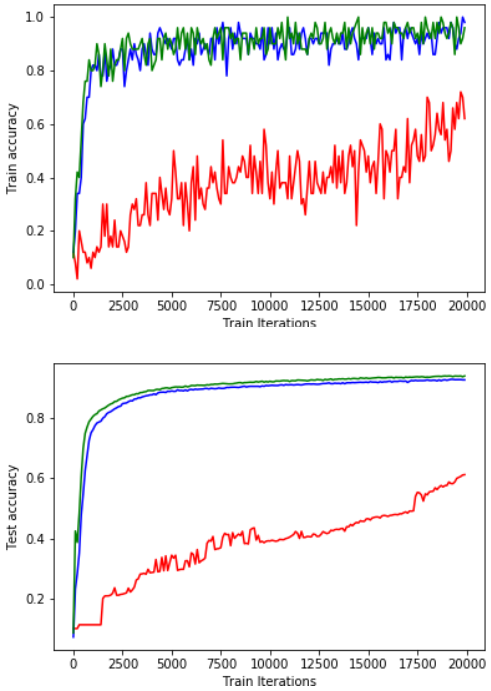

Figure 8. Experiment 6 - 8 layers with 1-Input, 1-Output,1-Softmax and 5-Hidden layers with 16 16,32,64,32 nodes at each hidden layer. Performance ELU is much better here than ReLU and Sigmoid.

improved drastically when we dropped two layers in the current experiment.

Inference - From the observation above it is now evident that with an increase in a number of layers bias-shift effect increased in ReLU based activation units thereby decreasing their accuracy. At the same time, Sigmoid Units faced a lot more vanishing Gradient Effect with an increase in a number of layers. This resulted in a decrease in accuracy. This is a direct indication of our initial predicament that ELU does mitigate both Vanishing Gradient and Bias-Shift effect.

3.2.3. Experiment 6.

Design - For experiment 6 we used a Network structure similar to shown in figure 2 with variance in a number of layers and how many nodes each layer has. The designed Network consists of 10 layers with 1-Input, 1-Output,1-Softmax and 5-Hidden Layers. Hidden layers have 16,16,32,64,32 nodes starting from the first hidden layer.the idea behind this experiment was to check if having a different number of nodes at each layer produce any substantially different results.

Inference - In such a network also, ELU performed substantially better than it's peers. Its classification accuracy was better than ReLU and far better than Sigmoid. Also, The training accuracy graph (figure 8) tells us a similar story.

3.2.4. Final Inference. After these series of Experiments, it can be inferred that ELU is better at performing Classification tasks with all kinds of layer and node configuration. Even they regularize faster then RelUS in lesser number of layers with less number of nodes. They outperform ReLU and Sigmoid in any large network.At the Same time, it was also proved that ELU mitigates the effect of Vanishing Gradients and Bias-Shift.

Figure 9. MNIST - (Pic Credit- [9])

4. Conclusion

Through the course of this Project, it was proved that ELU is a much better Activation function than ReLU and Sigmoid. Through mathematical analysis, it is proven that that there are two ways to reverse the Bias-shift effect. Either By using Unit Zero Mean Normalization better known as batch Normalization or by having Activation functions with Negative parts. ELU falls in second category. through our experiments we tried to validate this fact practically over a real dataset such as MINST. We analyzed various results and our deductions were satisfactory and as per expectations. More future work can be done with the ELU in

reconstruction analysis of Auto-Encoders and classification accuracy analysis on large Databases like CIFAR-100 [10] or ImageNet [11] with Convolutional Networks.

Acknowledgments

I acknowledge that, I was not involved in violations of any kind with respect to Academic Integrity Policy while working on this project. I am completely aware of consequences which would result from any such violations. I was made aware of this consequences from the discussion in the syllabus notes and the requirements of the project description. I also understand that, any such violation of the code of academic integrity will be reported to the dean for official actions. I also want to acknowledge here that, I have understood the requirements for this project and have implemented this project on my own.

I would also like thank Professor Li and Kevin for their dedicated support through out this Semester.

References

[1] Clevert, D.A., Unterthiner, T. and Hochreiter, S., 2015. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

[2] Amari, S.-I. Natural gradient works efficiently in learning. Neural Computation, 10(2):251276, 1998.

[3] Duda, R.O., Hart, P.E. and Stork, D.G., 2012. Pattern classification. John Wiley and Sons. Chapter -06

[4] Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudk, M. (eds.), JMLR WandCP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011), volume 15, pp. 315323, 2011

[5] Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the 30th International Conference on Machine Learning (ICML13), 2013.

[6] Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. CoRR, abs/1505.00853, 2015. URL http://arxiv.org/abs/1505.00853.

[7] http://yann.lecun.com/exdb/mnist/

[8] Kingma, D., and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. Chicago

[9] https://www.tensorflow.org/get_started/mnist/beginners

[10] https://www.cs.toronto.edu/ kriz/cifar.html

[11] http://www.image-net.org/