Exponential Linear Units (ELUs): Activation Function for Fast and Accurate Neural Network Training

Dhruv Arora and Priyanka Dubey

**Abstract**—We are exploring the advantages of “exponential linear unit” (ELU) for use as a hidden unit in deep neural networks. ELUs are like ReLUs but with some added advantages like having negative values which allows them to push mean unit activations closer to zero. This reduces the bias shift effect and ensures a noise-robust deactivation state. In our experiments, we observed that ELUs lead to the lowest cross-entropy loss on the training set and the best test set accuracy, as compared to other commonly used activation functions, in a 6-layer network on the MNIST dataset.

**Index Terms**—Deep Learning, Neural Network, ELU, ReLU, MNIST, TensorFlow.

—————————— ◆ ——————————

# 1 Introduction

T

o learn high level abstractions in domains like computer vision and natural language processing, we need deep architectures [1]. But training a deep network requires various design considerations like number of hidden layers, number of hidden units, choice of activation function, etc. The choice of hidden units is an extremely active area of research and does not yet have many definitive guiding theoretical principles [2]. Deciding which hidden unit to use is usually done through trial and error on a validation set. Rectified Linear Unit (ReLU) is the most popular choice for a hidden unit owing to its ease of optimization because of its similarity to linear units. The ReLU activation function is the identity for positive arguments and zero otherwise. But ReLUs cannot learn via gradient based methods on examples for which the activation is zero. The paper [3] proposes a new activation function, ELU, which combines the benefits of ReLU with improved learning characteristics compared to other units. Like ReLUs, the ELUs solve the problem of vanishing gradient (the gradients tend to become smaller and smaller in the earlier layers during backpropagation, slowing down learning) via the identity for positive values. Some key features of ELUs are-:

**Negative Values.** ELUs have negative values which pushes the mean of the activations closer to zero. The paper [3] shows with the help of a mathematical proof that mean activations that are closer to zero **enable faster learning** as they bring the gradient closer to the natural gradient. It also helps in **alleviating the phenomenon of *bias shift*** (or *covariate shift*). ELUs saturate to a negative value with smaller inputs and therefore reduce the forward propagated variation and information. The hyperparameter ‘α’ controls the level of saturation.

————————————————

1. Dhruv Arora is a Master’s student in Computer Science at Arizona State University, Tempe, AZ. E-mail: darora8@asu.edu
2. Priyanka Dubey is a Master’s student in Computer Science at Arizona State University, Tempe, AZ. E-mail: pdubey2@asu.edu.

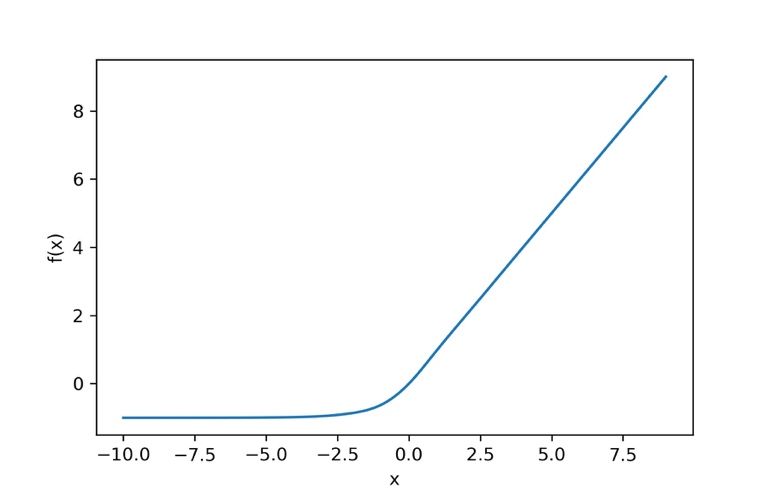
**Robust to Noise.** Since the ELUs saturate to a negative value for smaller arguments, it decreases the variation of the units if deactivated and so the precise deactivation argument is not relevant. This can help detect the presence of a phenomenon but does not quantitatively model the degree of their absence, making it more robust to noise. Only the activated units carry the relevant information.

The authors [3] have tested the performance of ELUs for both unsupervised and supervised learning on the MNIST, CIFAR-10, CIFAR-100 and ImageNet datasets. For a fully-connected deep network, ELUs with α=1.0 perform extremely well: the median value for the hidden units have a smaller median throughout the learning and the training error decreases more rapidly than for other networks. ELUs was among the top 10 results reported for CIFAR-10 with a test error of 6.55%. Also, they performed the best on CIFAR-100 with a test error of 24.28%, which is the best reported result for the dataset.

For our experiments, we have used a six-layer network for training on the MNIST dataset. Each hidden layer had 20 units and we trained 4 different networks using Sigmoid, ReLU, ELU and LeakyReLU as the activation functions. To be consistent with results obtained by the authors of the ELU paper [3], we used the same hyperparameters like the number of epochs, learning rate, mini-batch size, initialization technique and the value of α in ELU and LReLU. TensorFlow [4] was used for neural network training. No GPU was used for this experimentation. The ELU network had the lowest cross-entropy loss on the training set among all the networks. The test accuracy was also highest with the ELU network although the ReLU network was comparable. Strangely enough, the ELU network trained ten seconds slower (average over 10 iterations of 300 epochs each) than the ReLU netowork which proved to be the fastest among all.

## 2 Exponential Linear Units

The *expoenential linear unit* with α > 0 is given as: *f(x) = x if x >0 or α (exp(x)-1) if x<=0*. The parameter α controls the value to which an ELU saturates for negative input values (see Fig. 1).



,m,.m,.,m…………….m,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

fnklnd

Fig1. The Exponential Linear Unit (α=1)

In deep networks, the gradients tend to get smaller as learning progresses especially in the starting layers. This means that the units in the earlier layers learn much more slowly than the ones in the later layers. This is also known as the *vanishing gradient problem*. To overcome this problem, we try and use hidden units whose derivative is not contractive, so that while using the chain rule during backpropogation the gradients do not disappear. The positive part of an ELU is the identity and hence the derivative is one which help alleviates this problem. Also, as can be seen from Fig.1 the function has negative values which pushes the mean of the activations closer to zero. Mean activations that are closer to zero enable faster learning as they bring the gradient closer to the natural gradient [3]. The saturation parameter *α* ensures that the variation and the information which is propogated to the next layer decreases. Therefore, the representation is both noise-robust and low-complex [5]. Also, by bringing the mean activations closer to zero the bias shift is corrected. One example of the presence of bias shift in a neural network is when it is trained to identify cats of black color but during testing, if an image of a different color cat is provided the network fails to correctly classify it as a cat. To resolve this issue, the mean and variance of hidden units in each layer should not vary much from each other. Like batch normalization whch is commonly used for this purpose, ELUs will also aid in removing the bias shift. ELUs thus help in propogating the gradients well and computing interesting features because of their non-linearity. The authors of the ELU paper [3] have shown that as compared to other activation functions, the median of the units’ average activation was smallest throughout the training process. The median varies much more in ReLU networks indicating that ReLU networks continuously try to correct the bias shift while this is much less prominent in the case of ELUs.

**3. Experimental Setting and Dataset**

To assess the performance of various activation functions we needed a neural network of a decent size. In our experiments, we have implemented a six-layer network with the last layer being a softmax, which is often used when we wish to represent a probability distribution over a discrete variable with n possible values. It is a generalization of the sigmoid function, which is used to represent a probability distribution over a random variable. The softmax function is defined as:

*softmax(z)i = exp (zi) / Σj exp(zj)*

The discrete variable in our study is the digit to be classified ranging from 0 to 9 in the MNIST [6] dataset. The data is comprised of 28 X 28 gray images in 10 classes (0-9), 55k train and 10k test. A sample image is shown in Fig2.

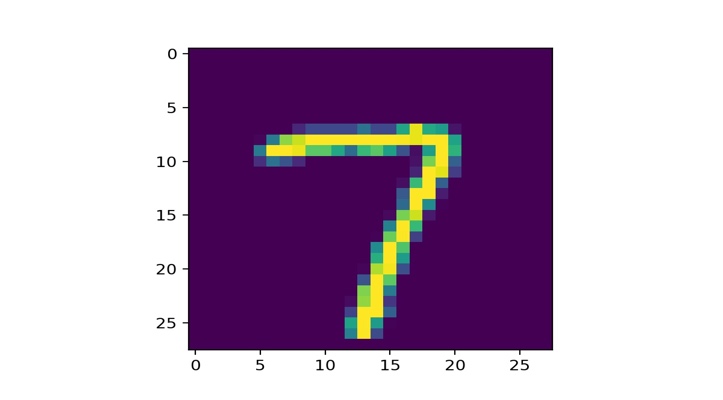


Fig2: MNIST image of a ‘7’

The network is built using Tensorflow [4] which is a powerful library for numerical computation. The common usage for TensorFlow programs is to first create a graph and then launch it in a session. Our network contains 20 units in each hidden layer. The weights have been initialized per the He [7] technique, which helps in ‘breaking the symmetry’. Four different networks were trained: Sigmoid, ELU (α=0.1), ReLU and LReLU (α=0.1). ‘Leaky ReLUs’ (LReLUs) replace the negative part of the ReLU with a linear function [8]. Each network was trained for 300 epochs by stochastic gradient descent with a batch size of 64 and a learning rate of 0.01. In every epoch, a new random minibatch [9] was created to feed in the network. The hyperparameter values were kept in conjunction with the ELU paper [4]. The training and test eror was recorded for each network along with the running time. Activation values for each hidden layer after every epoch of training were also analyzed. This helped us to verify the bias shift correction present in ELU networks which helps speed up learning. With the aim of further improving the performance and to see the effect of regularization on ELU, dropout [10] with a probability of 0.5 was used. The results are tabulated and discussed in detail in the next section. Code for this work can be found at: <https://github.com/dhruvarora93/Deep-Learning-with-ELU>.

**4. Results**

The training and testing set accuracy is tabulated in Table 1. ELU outperformed the other networks in both aspects.

|  |  |  |
| --- | --- | --- |
|  | Training Accuracy | Test Accuracy |
| Sigmoid | 95.083 % | 93.68 % |
| ReLU | 98.065 % | 95.36 % |
| **ELU** | **98.798 %** | **95.75 %** |
| LReLU | 97.610 % | 94.84 % |

Table 1: Accuracy Comparison

The training set cross-entropy loss was the minimum in ELU networks as can be seen in Fig. 3. All lines stay flat after epoch 60.

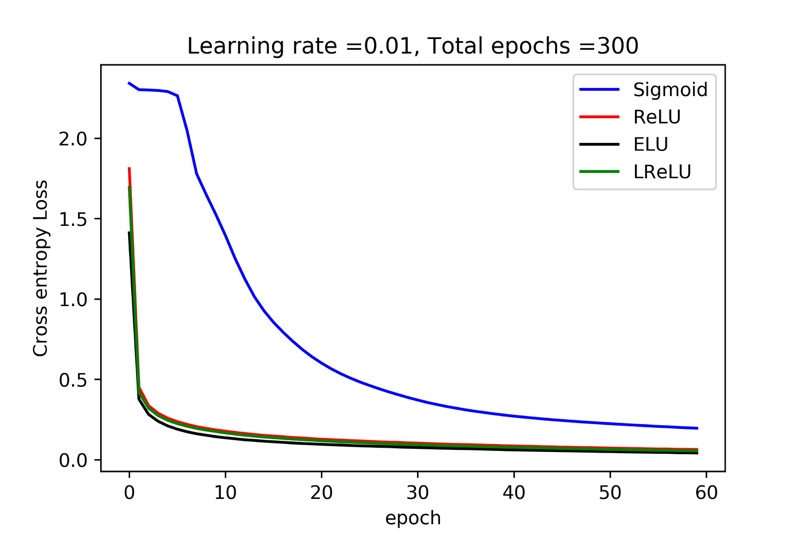


Fig.3: Traning Set Cross-Entropy Loss

It is evident from Fig. 3 that the training error of ELU networks decreases much more rapidly than for other networks. Thus, we can indeed colclude that they possess a superior learning behavior compared to other activation functions. The running time for training of each network can be seen in Table 2. ReLU network trained the fastest.

|  |  |
| --- | --- |
| **Network** | **Training Time (in seconds)** |
| Sigmoid | 601.384 |
| **ReLU** | **598.723** |
| **ELU** | **608.894** |
| LReLU | 761.337 |

Table 2: Running Time

Also, after each epoch we calculated the units’ average activations on the same minibatch used for training (see Fig. 4). We see that as training progresses, the activation values increase for every network but since ELU networks try and bring the mean of activations closer to zero, they have the lowest set of values among all networks (LReLU were also comparable), which is a desirable property to have in deep networks.

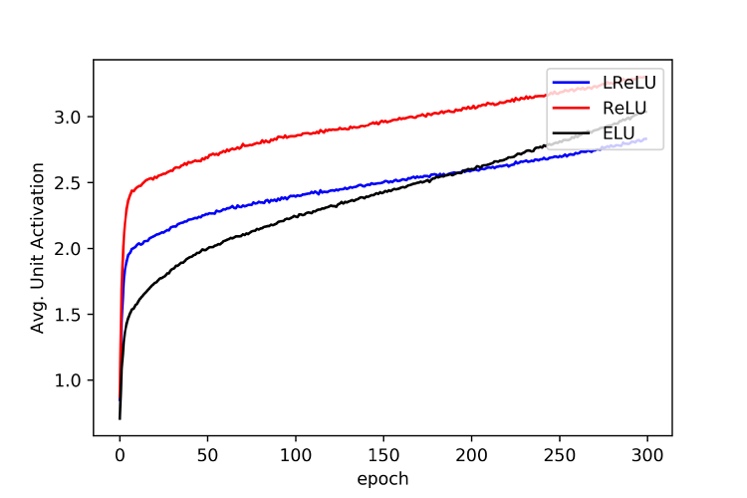


Fig. 4: Average Unit activation

**4.1.1. Dropout**

Dropout [10] is a very powerful regularization technique whose main idea is to randomly drop units from a network during training. We tried to use dropout on ELU networks to further improve improve performance on the test set. We started with a probability of 0.6; the hyperparameter which controls the probability of dropping a neuron. **Surprisingly, the training and test accuracy dropped drastically to 76 % and 76.59 % respectively**. After this, dropout was applied to alternate hidden layers starting from the first one and the number of epochs were increased to 500. No improvement was found in this case either. Reducing the probability to 0.3 and keeping the number of epochs at 300 was of no improvement either. Hence, droput did not prove to be a useful regularizer for our ELU network. Among the many reasons possible for this, we strongly felt that the main reason was the less number of hidden neurons (20) in each layer.

**5. Conclusion and Future Work**

We have reviewed and explored the advantages of exponential linear units for use as activation function in deep architechtures using a six-layer structure created in Tensorflow. ELU network performed the best on the dataset compared to any other powerful activation function like ReLU, which validates their use in image recognition tasks. No wonder the authors of the paper [4] achieved groundbreaking results on many vision datasets. The negative values of ELU constantly try and push the mean activations closer to zero enabling faster and efficient learning. We would like to use ELU in convolution networks as they can prove to be a real time saver. Further, we would like to extend our network to more layers and hidden units to work on datasets like the CIFAR -10 and SVHN. Due to a computational constraint, we could not build a bigger network, but given the right resources we would like to further work and explore computer vision applications where ELU can prove to be a better alternative. Also, we would like to delve deeper in research involving activation functions for deep architechtures.

**6. References**

1. Bengio, Y. (2009). Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2, 1–127. Also, published as a book.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
3. Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." arXiv preprint arXiv:1511.07289 (2015).
4. Software available from <https://www.tensorflow.org>.
5. Hochreiter, S. and Schmidhuber, J. Feature extraction through LOCOCODE. Neural Computation, 11(3): 679–714, 1999.
6. <http://yann.lecun.com/exdb/mnist/>
7. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).
8. Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013, June). Rectifier nonlinearities improve neural network acoustic models. In Proc. ICML (Vol. 30, No. 1).
9. <https://www.coursera.org/learn/neural-networks-deep-learning/>
10. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research, 15(1), 1929-1958.