# Evaluation of Activation Functions on Image Classification of Neural Networks

**Wei Zheng**
Department of Computer Science
999986261
zhengw14@cs.toronto.edu

**Jianing Guo**
Department of Computer Science
999489192
scarlettguo@cs.toronto.edu

## Abstract

The choice of activation function may have a significant impact on the training speed and prediction accuracy of neural network classifiers. Many new activation functions claim to hold performance advantages over commonly used activation function ReLU. In this study, we attempt to verify the results by conducting exploratory experimentation on various activation functions including Swish, ReLU, Leaky ReLU, ELU, and Tanh. We conclude desired properties of an activation function. Additionally, we propose a new activation function based on characteristics from Swish and ReLU. Experiments were performed using models of varying complexity including fully connected networks, simple CNNs, ResNet, and WRN. We use MNIST, SVHN, CIFAR-10 and CIFAR-100 in our experiments. Our findings show that for network dimensions and problem difficulties tested, Swish did not show any considerable advantage over ReLU while our newly proposed activation function is competitive with ReLU in several cases.

## 1 Introduction

Activation functions play an important role in deep neural networks. Deep neural networks using a non-linear activation function can approximate any complex function [9], but in finite networks, the choice of activation function affects the learning dynamics and the network's expressive power [2]. Without a non-linear activation function, multilayer neural networks have the same expressive power as single hidden layer neural networks. The most commonly used activation function is the Rectified Linear Unit (ReLU) [18, 4]. ReLU enables supervised training on deep networks [16].

Recently, Google researchers found a new activation function called Swish [21]. It was found that Swish matches or outperforms ReLU in many challenging experiments [21]. Inspired by the finding, we would like to validate the results by comparing Swish with ReLU and few other activation functions. We also propose a new activation function and performed experiments to compare it with the most popular activation functions.

Section 2 presents related works in the field of evaluating and finding activation functions. Section 3 introduces the new activation function we proposed and other activation functions used in the experiments. Section 4 provides evaluation of the activation functions on 4 datasets: MNIST, SVHN [19], CIFAR10 and CIFAR100 [15]. We conclude the paper on section 5. In section 6, we discuss limitations and future works of this paper.

## 2 Related Works

Ramachandran uses automatic search techniques to discover new activation functions [21], from which Ramachandran found Swish $f(x) = x * sigmoid(x)$ has the best performance over all other functions [21]. Swish units improve the top-1 classification accuracy on ImageNet by 0.9% for

Mobile NASNet-A[21]. The searches discovered activation functions that utilize periodic functions, and most of the existing activation functions [18, 17, 14, 7] do not incorporate periodic functions like $sin$ and $cos$. Our proposed activation function was inspired by the performance of ReLU. However, we focus on simple tasks and experiments that do not require massive hardware resources.

Many works have presented new activation functions to replace ReLU by solving the dead ReLU problem [3, 7, 17, 26]. However, none of the proposed replacements thus far have gain widespread adoption [21]. Leaky ReLU [17] added a small slope in the negative region. ELU [3] has negative values which allows it to push mean unit activations closer to zero like batch normalization.

Xu [26] conducted evaluations for the different types of rectified activation functions in convolution neural network including: ReLU, Leaky ReLU, PreLU and RReLU [26]. Xu showed that incorporating a non-zero slope for the negative part in rectified activation units could consistently improve the results [26]. This work only focuses on the rectified activation functions and does not incorporate other activation function like Tanh and maxout [6].

While this work focus on point-wise activation function, there are other many-to-one non-linear activation used in deep networks [21]. Maxout [6] is one of the popular many-to-one non-linear function. A maxout feature map construct by taking the maximum across k pool across channel with spatial location [6]. Maxout networks learn not just the relationship between hidden units, but also the activation function of each hidden unit [6]. Maxout activation function can approximate arbitrary convex functions like ReLU and quadractic equations [6].

## 3 Activation Functions

Base on reading the papers [3, 4, 12, 14, 18, 21, 22, 26] we found following are desired properties of an activation function.

1. *Non-linearity* - key requirement of activation function, it maintains expressiveness of multi-layer neural networks [9].
2. *Almost Everywhere Differentiable or Differentiable* - being differentiable is required for stochastic gradient descent optimization. A finite number of non-differentiable points does not affect the optimization result [5].
3. *Simple Calculation and Few parameters* - Any non linear function can be an activation function. However, the number of activation function calculations is correlated with number of neurons in the network so a simple non linear function is suitable for being used as the activation function especially in deep networks [21]. Therefore, simple functions like ReLU are more popular than activation functions using the exponential function like tanh, ELU and sigmoid [18, 21].
4. *Avoid Saturation* - Saturation means that the gradient vanishes in an interval. The most common case is sigmoid which has close to 0 value for its derivative at large values. Sigmoid saturates and kill gradients, it needs careful initialization to avoid saturation [4]. ReLU has 1 gradient at positive region and 0 gradients at the negative region. For the negative region, it would be in saturation state which we call dying ReLU. Leaky ReLU [17] and PReLU [7] were developed to solve the dying ReLU problem.
5. *Monotonic* - monotonic activation functions make the gradient descent stable and therefore increase the speed of model convergences.
6. *Restricted range* - restricted range of activation like tanh (between -1 and 1) make the neural network stable even if the input values are large however it contradicts with the gradient vanishing problem. Restricted output range will limit the expressiveness of the layer.
7. *Approximately equal to f(x)=x* - When the activation function does not approximate the identity function need the origin, it needs special care for the initialization values [4]. This property also makes the gradient calculation easier. ReLU has this property in the positive region. This property is supported by the findings in the Swish paper [21] that most of the top novel activation functions found by the search have a linear portion.
8. *Normalization* - Normalization for activation has recently been introduced on the SELUs paper [14]. The key idea is activation functions close to zero mean and unit variance will

make the network layers converge towards zero mean and unit variance [14]. This property will enable training on deeper networks. Batch Normalization uses the similar idea [11].

We included the following activation functions in the evaluation section: ReLU, Leaky ReLU, ELU, Tanh, and Swish. Figure 1 shows the activation and Table1 give the activation functions' formula. Sigmoid has an average value of 0.5 and Tanh is similar to sigmoid with average value 0 and range between 1 and -1. Sigmoid and Tanh were popular thanks to the limited ranges [5]. The saturation property of Sigmoid and Tanh makes deep neural networks difficult to train [18]. However, modern techniques like Batch Normalization can effectively relieve the saturation problem [11]. Tanh is a substitute of sigmoid in most cases. Tanh and Sigmoid are usually used for the activation function in the output layers because of their limited range.

The introduction of ReLU made supervised training of deep networks possible [18, 4]. ReLU has hard non-linearity and non-differentiability at zero, creating sparse representations with true zeros, which seems remarkably suitable for naturally sparse data [4]. Sparsity in the network disentangle the factors explaining the variations in the data. ReLU has most of the desired properties defined above. However, ReLU needs careful initialization for learning rate and weights in order to avoid saturation. ELU [3] and Leaky ReLU [17] solve the dying ReLU problem by adding a slope to the negative region.

Swish [21] is very similar to ReLU with the main difference in the negative domain. We believe performance gain of Swish over ReLU in the experiments [21] has to do with the saturation problem on the negative domain. Swish has some interval in the negative region that has a non-zero gradient. Average values of most of the initial weights are usually zero so half of the initial weights are in the negative domain [5]. Input from the batch normalization [11] is also close to zero mean.

In addition to the above activation functions, we propose a new activation that combines the idea of Swish and ReLU. The function 1 is the same as ReLU in the positive region and similar to Swish in the negative domain. The function definition is defined in equation 1 and its first derivative is defined in equation 2. Figure 2 shows the derivative of the new activation function and Swish. The function calculation and gradient calculation are simpler compared to Swish. To the best of our knowledge, this is the first paper to use this activation function.

Equation of the activation function

$$f(x) = max(x, xe^{-|x|}) \text{ equivalently } f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ xe^{x}, & \text{otherwise} \end{cases} \tag{1}$$

First derivative of the activation function

$$f(x) = min(1, (1+x)e^{x}) \text{ equivalently } f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ (1+x)e^{x}, & \text{otherwise} \end{cases} \tag{2}$$

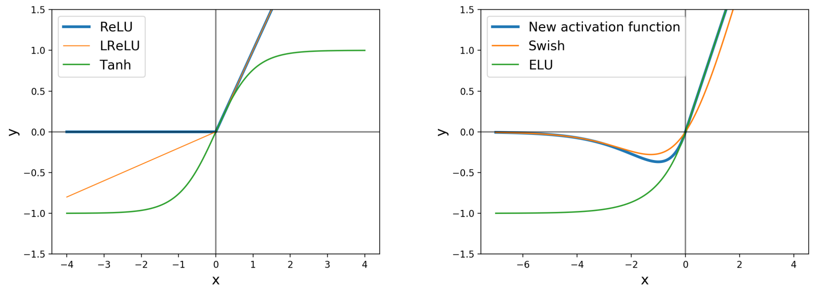


Figure 1: Activation functions

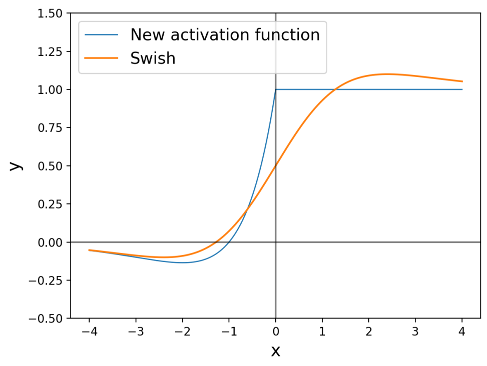


Figure 2: Derivative of Swish and new activation function

| Activation Function | Formula | Note |
|---|---|---|
| ReLU | $f(x) = max(0, x)$ | |
| Leak ReLU | $f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases}$ | $0 < \alpha < 1$ |
| ELU | $f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(exp(x) - 1), & \text{otherwise} \end{cases}$ | $\alpha > 0$ |
| Tanh | $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ | |
| Swish | $f(x) = x\sigma(x) = \frac{x}{1+e^{-x}}$ | |
| New Activation | $f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ xe^{x}, & \text{otherwise} \end{cases}$ or $f(x) = max(x, xexp(-\lvert x \rvert))$ | |

Table 1: Activation functions

## 4 Evaluation

To run our experiments, we used the GPU server available for CS students at the Computer Science Department of University of Toronto. Due to the limitation of hardware resources available, we did not include the ImageNET dataset in the experiments. We used ReLU as the baseline and compare the performance of other activation functions with it. For section 4.1 to 4.3.1 we use Leaky ReLU with $\alpha$=0.2, the default value in Tensorflow[1]. A similar $\alpha$ value is suggested by Xu [26]. For section 4.3.2 we use Leaky ReLU with $\alpha$=0.01, the default value for PyTorch [20]. In section 4.3.2 we compare the performance of two $\alpha$. For ELU, we set the $\alpha$ to 1 as the original paper suggested [3].

### 4.1 MNIST on Fully Connected Network

The dataset used in this experiment is the MNIST dataset from Tensorflow [1]. Each record of MNIST is a 28x28 black and white image of a hand written digit from 0 to 9 [1]. The MNIST data is split into three parts: 55,000 instances of training data, 10,000 instances of test data, and 5,000 instances of validation data [1]. We first evaluated the activation functions on fully connected networks available on GitHub [13]. We used a standard fully connected 2-layers neural network with 500 neurons in the hidden layer [13]. The learning rate base is 0.8 with exponential learning rate decay at 0.99 [13]. The models were trained for 10000 steps with a regularization rate of 0.0001. Each batch size is 100 without Batch Normalization [11]. Then, we use the network to 11-layers of fully connect neural network with the first 7 hidden layers having 500 neurons on each layer and 300,200,100 neurons per layer on the last hidden 3 layers [13]. In the 11-layer network, the learning rate base is decreased to 0.008.

The result of this experiment is summarized in Table 2. Each of the entries in Table 2 is the median of 5 runs. Validation accuracy and test accuracy for the activation functions are similar for the 2-layer network. LReLU and ELU are slightly worse. For the 11-layer network, all activation function experienced small accuracy decrease because the hyper-parameters are not optimized for 11-layers. Accuracy of Tanh and the new activation function decreased significantly. LReLU and ELU suffer less accuracy decrease. Swish and ReLU have the similar performance on the fully connected network. In the original paper of Swish [21], Swish performs better than ReLU on the over 40 layers of fully connected neural network. Each of the layers has 1,024 neurons.

| Activation Function | 2-layer Validation | 11-layer Validation | 2-layer Test | 11-layer Test |
|---|---|---|---|---|
| ReLU | 98.44 | 97 | 98.35 | 96.72 |
| Leak ReLU | 97.9 | 97.1 | 97.88 | 96.86 |
| ELU | 97.82 | 96.94 | 97.64 | 96.7 |
| Tanh | 98.34 | 96.8 | 98.13 | 96.41 |
| Swish | 98.46 | 97.22 | 98.22 | 96.75 |
| New Activation | 98.54 | 96.68 | 98.33 | 96.47 |

Table 2: MNIST validation accuracy and test accuracy in percentage

## 4.2 SVHN

The Street View House Numbers (SVHN) dataset is a collection of image clippings of house numbers from Google Streetview. The SVHN dataset contains 73257 training images, 26032 test images, and 531131 extra images for training or validation. There are two collections associated with SVHN and a distinct problem for each collection. The harder problem involves reading consecutive digits from images of house numbers in their entirety. The simpler problem involves classifying single digits (0-9) in 32 by 32 RGB space. This is similar to MNIST but more difficult due to the presence of distracting elements in the images [19].

A convolutional neural network [25] on GitHub made using TensorFlow with two convolution layers and two fully-connected layers were used to evaluate the performance of the various activation functions. SVHN images were transformed into the grayscale colourspace for simpler processing. Each model was trained for 100 epochs using a base learning rate of 0.05, decay of 0.96, and batch size 128 [25].

The results of this experiment are summarized in Table 3.

| Activation Function | Testing Accuracy at Final Epoch |
|---|---|
| ReLU | 94.72 |
| Leaky ReLU | 94.13 |
| ELU | 93.95 |
| Tanh | 91.78 |
| Swish | 92.84 |
| New Activation | **94.94** |

Table 3: Summary of test accuracy for each activation function on the SVHN 32x32 dataset

For this experiment on a simple CNN model, our newly proposed activation function outperformed the baseline ReLU by a slim margin, while outperforming Swish by a significant margin. In fact, the relatively poor performance of Swish is a surprise since in the Swish paper [21], it was claimed that Swish holds an advantage over ReLU in deep networks. It is possible that the advantage of Swish over ReLU (and its derivatives) only become apparent in deeper networks. Thus, further experimentation was done on more complex models and harder classification problems.

## 4.3 CIFAR

We use the CIFAR-10 and CIFAR-100 dataset [15] on the ResNet [8] and Wide Residual Network (WRN) [27] models to perform further experimentation. The CIFAR-10 dataset consists of 60000 32x32 coloured(RGB) images arranged into 10 classes [15]. The CIFAR-100 dataset consists of 60000

32x32 colour images in 100 classes [15] We use the Tensorflow [1] official model implementation of ResNet. For WRN [27], we use the authors' PyTorch [20, 27] implementation.

#### 4.3.1 ResNet

Our implementation of the Deep Residual Network model is based on Tensorflow's official ResNet implementation [8, 1]. However, since the official model only supports CIFAR-10 and ImageNet as input datasets, modifications were added to the code in order to accommodate CIFAR-100. Additionally, the code was augmented to allow other activation functions to be used which can be specified at run time.

For CIFAR-10, we used a 32-layer deep ResNet model with weight decay set to $0.0002$ and momentum set to $0.9$. The base learning rate was set $0.1$ and decays by a factor of $0.1$ at epoch 80 and epoch 120. The model was trained for 160 epochs int total for each activation function. The models use Batch Normalization [11]. Accuracy tensors were logged in order to visualize the change in accuracy over the course of training for each of the six models, see Figure 3 illustrates the change in training accuracy as training progresses.

| Activation Function | CIFAR10 Test Acc. | CIFAR100 Test Acc. |
|---|---|---|
| ReLU | **92.31** | 67.95 |
| Tanh | 89.70 | 63.51 |
| Leaky ReLU | 91.54 | **68.73** |
| New Activation | 92.30 | 67.95 |
| Swish | 92.30 | 68.26 |
| ELU | 90.48 | 67.81 |

Table 4: ResNet CIFAR10 and CIFAR100 test accuracy in percentage

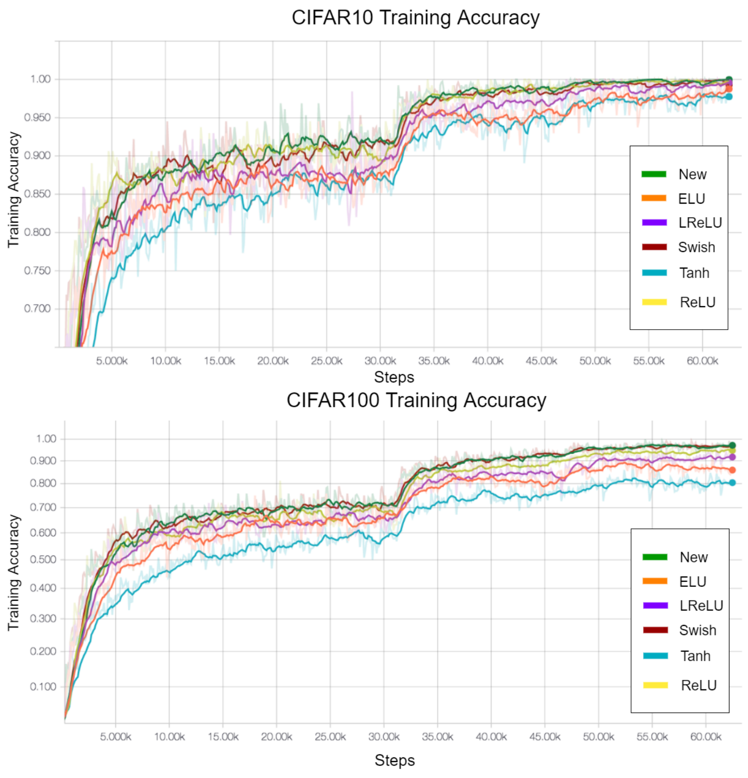


Figure 3: ResNet-32 CIFAR10 and CIFAR100 Training Accuracy

Table 4 summarizes the stabilized testing accuracies for the CIFAR-10 and CIFAR-100 experiments the 32-layer ResNet model. From the results, it is evident that there is no clear gap in performance

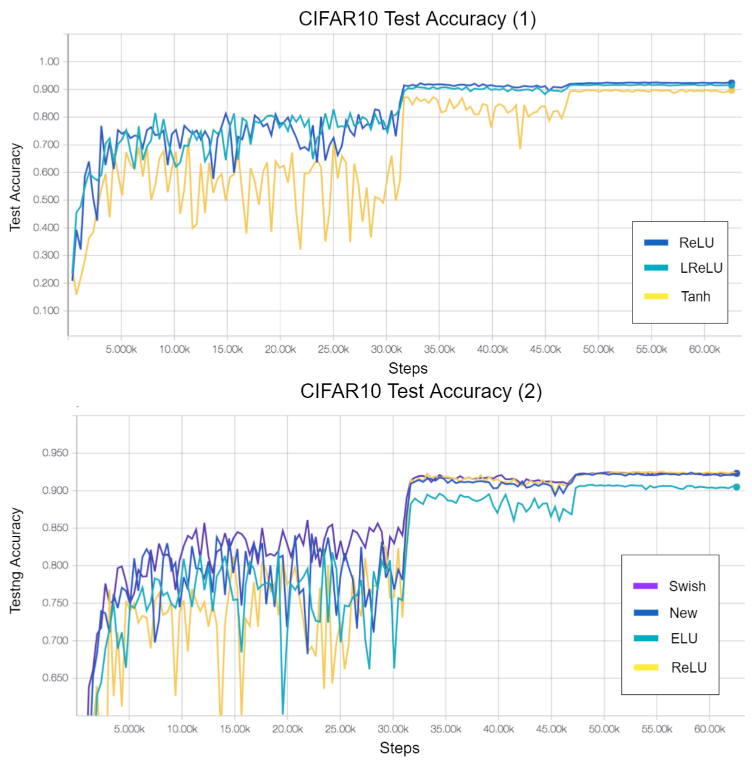


Figure 4: ResNet-32 Test Accuracy CIFAR10

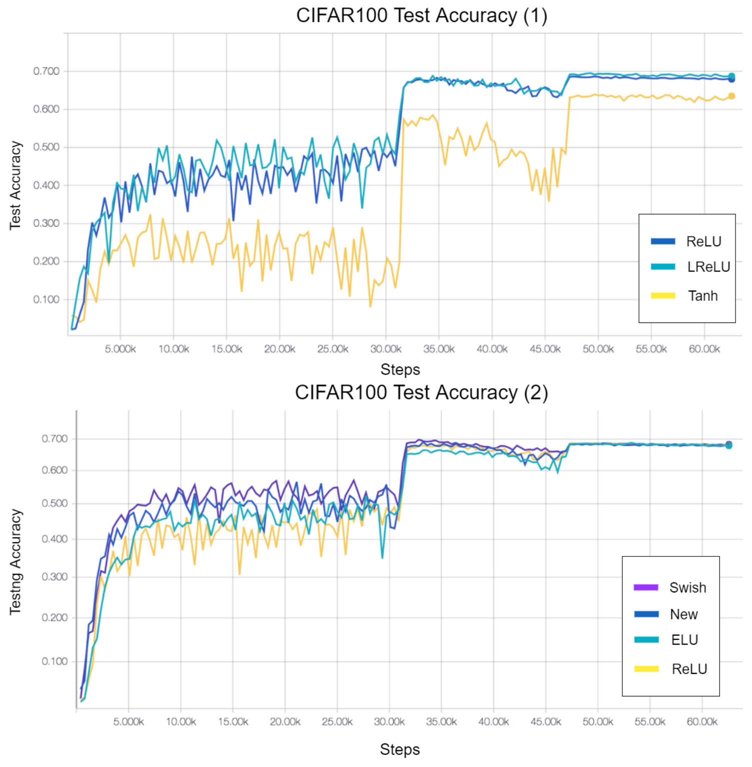


Figure 5: ResNet-32 Test Accuracy CIFAR100

between 4 out of 6 activation functions tested: ReLU, Leaky ReLU, Swish, and the proposed New Activation Function. ELU lags in performance for CIFAR-10, but is competitive with the top performers in CIFAR-100. In general, Tanh is consistently the worst performer across all trials, while ReLU, Swish, Leaky ReLU and New Activation remain the top performers. As can be seen in Figure 4, The gap between ELU and the other functions only widen as classification difficulty is increased.

To further observe performance differences between the six activation functions under consideration, we re-ran the experiment on the CIFAR-100 dataset using a 14-layer as well as a 50-layer ResNet model. The increasing network depth and steeper classification difficulty were selected to magnify any differences in performance. Table 5 presents the results from the experiments on ResNet-50.

| Activation Function | Resnet-14 | ResNet-32 | ResNet-50 |
|---|---|---|---|
| ReLU | 65.13 | 67.95 | **69.80** |
| Tanh | 59.93 | 63.51 | 63.76 |
| Leaky ReLU | 64.83 | **68.73** | 69.20 |
| New Activation | 65.02 | 67.95 | 69.76 |
| Swish | **65.24** | 68.26 | 69.30 |
| ELU | 65.20 | 67.81 | 67.92 |

Table 5: ResNet CIFAR100 test accuracy on different layer numbers in percentage

Evidently, despite being competitive amongst the top scoring functions, the Swish activation did not display any clear advantage over ReLU nor Leaky ReLU. In fact, increasing network size only boosted the relative performance of ReLU over the other functions. On the other hand, our newly proposed activation function held up remarkably well to the competition, staying competitive with ReLU on all ResNet sizes that were tested, although never holding an advantage.

### 4.3.2 Wide Residual Network

For Wide Residual Network we use the same hyper-parameters described in the original paper [27]. We use WRN-16-4, which is widening parameter of 4 and 16 layers. The model has over 2.77 million parameters. For comparison, ResNet-110 has 1.7 million parameters. However, the training parameters for WRN-16-4 is roughly same as ResNet-32. The networks are trained for 200 epochs with initial learning rate 0.1. The learning rate will decay by multiplying 0.2 at 60, 120 and 160 epochs [27]. The model uses SGD with Nesterov momentum and cross-entropy loss [27]. The weight decay is set to 0.0005, dampening is 0, momentum is 0.9 and minibatch size is 128 [27]. The models use Batch Normalization. [11]. Please note that $\alpha$ parameter for Leaky ReLU use in this WRN section is 0.01. We are going to compare the performance between $\alpha$=0.2 and $\alpha$=0.01 at the end of this section.

ReLU and Leaky ReLU have the best performance on test accuracy on the CIFAR-10 and CIFAR100 datasets. Leaky ReLU has a small advantage over ReLU. Our new activation function performs slightly better than Swish. However, there is 1%-2% difference from the baseline ReLU. Tanh and ELU have the worst performance which significantly differs from the baseline. Table 6 gives the test accuracy for the WRN-16-4 model. Figure 6, 7 and 8 give the train accuracy and test accuracy for CIFAR-10 and CIFAR-100 datasets during the training. Training loss and test loss graphs are reported in the appendix section 7. From Figure 7 and 8, Swish has similar performance with the baseline ReLU on the first 120 epoch of the training. Specifically, on CIFAR-100, Swish has better performance over the baseline during the first 60 epochs. Also, from the same figures, we can see that tanh is unstable during the training. It is hard for tanh to converge in the training which matches our expectation as we described in the previous section. Overall, Leaky ReLU and ReLU have the best performance in WRN-16-4 model on CIFAR datasets.

Training loss and test loss graphs are reported in the appendix section 7.

Table 7 shows the test accuracy for the WRN-16-4 model on CIFAR-100 given 128, 256, and 512 batch sizes. All models suffer decreases in accuracy when the batch size was increased to 512. Leaky ReLU and ReLU suffer the most. However, Leaky ReLU and ReLU still have the best performance, but the difference between the baseline and other activation function become smaller when the batch size increases.

| Activation Function | CIFAR10 Test Acc. | CIFAR100 Test Acc. |
|---|---|---|
| ReLU | 95.08 | 76.5 |
| Tanh | 89.92 | 68.62 |
| Leak ReLU | 95.34 | 77.17 |
| New Activation | 93.81 | 75.01 |
| Swish | 93.63 | 74.45 |
| ELU | 91.39 | 71.51 |

Table 6: WRN CIFAR10 and CIFAR100 test accuracy in percentage

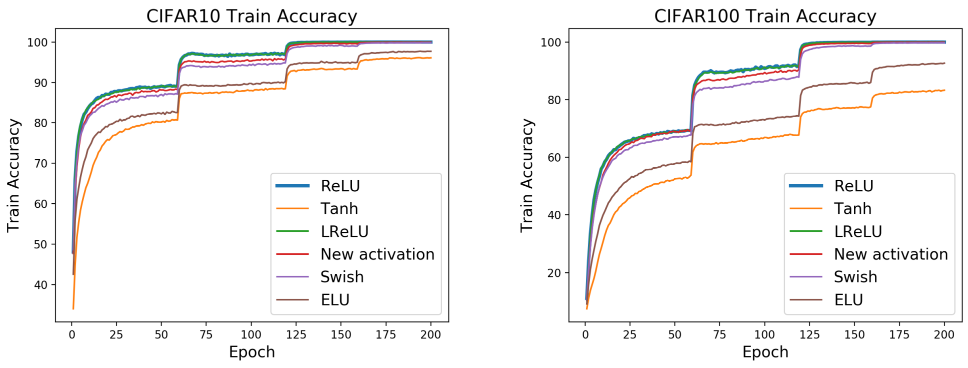


Figure 6: WRN Training Accuracy CIFAR10 and CIFAR100 batch size=128

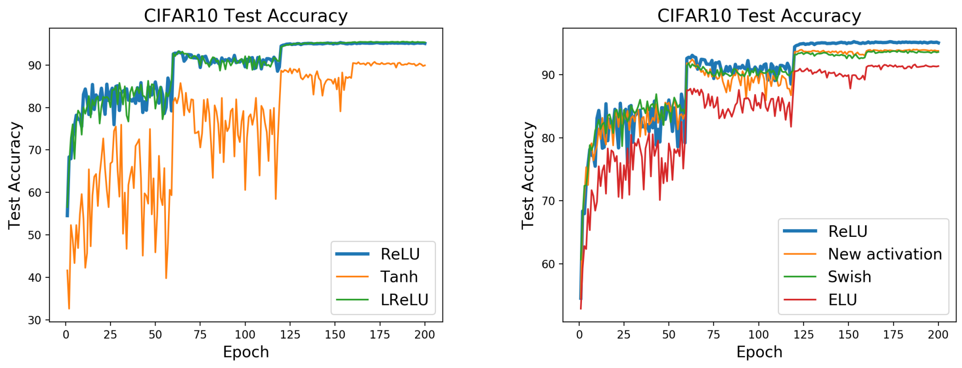


Figure 7: WRN Test Accuracy CIFAR10 batch size=128

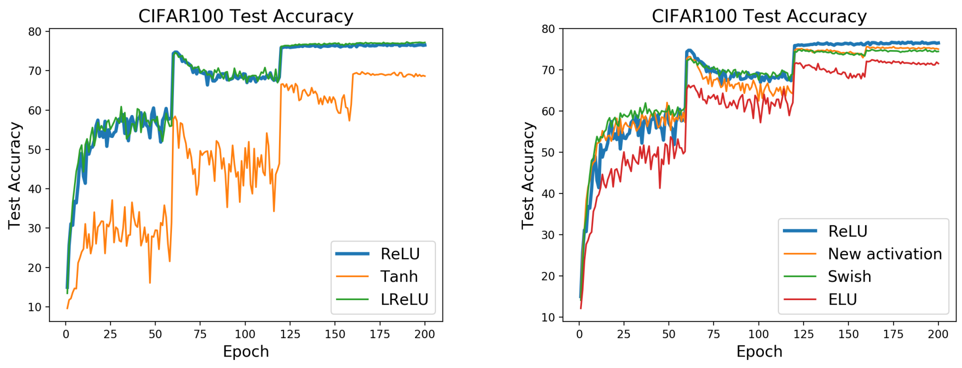


Figure 8: WRN Test Accuracy CIFAR100 batch size=128

| Activation Function | CIFAR100 | Test Acc. | |
|---|---|---|---|
| Batch size | 128 | 256 | 512 |
| ReLU | 76.5 | 75.88 | 74.18 |
| Tanh | 68.62 | 68.34 | 67.6 |
| Leak ReLU | 77.17 | 76.22 | 74.29 |
| New Activation | 75.01 | 74.7 | 73.4 |
| Swish | 74.45 | 74.49 | 73.35 |
| ELU | 71.51 | 72.07 | 71.3 |

Table 7: WRN CIFAR100 test accuracy on different batch sizes in percentage

We compared Leaky ReLU with $\alpha$=0.01 and Leaky ReLU with $\alpha$=0.2. Figure 9 shows training and test accuracy over epochs for both setups. We found that Leaky ReLU with $\alpha$=0.01 is consistently more accurate. According to Table 6, Leaky ReLU with $\alpha$=0.01 has 95.34% test accuracy for CIFAR-10 and 77.17% for CIFAR-100. Leaky ReLU with $\alpha$=0.2 has 94.64% test accuracy for CIFAR-10 and 76.37% for CIFAR-100. This result contradicts what Xu's result [26] in the evaluation paper.

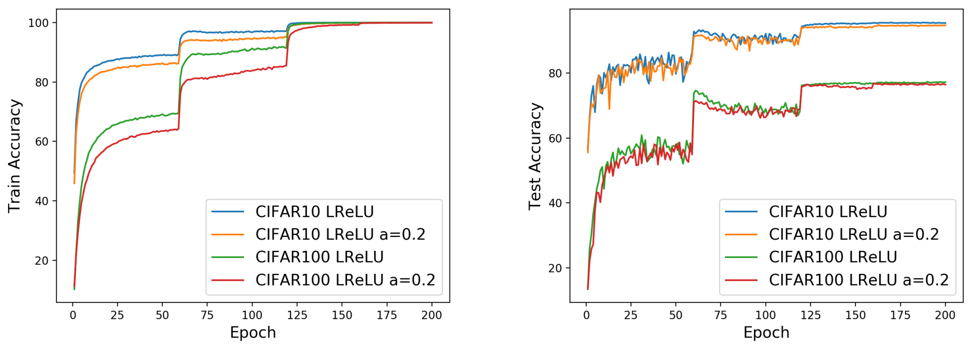


Figure 9: WRN Leak ReLU $\alpha$=0.01 and $\alpha$=0.2

## 5 Conclusion

In this paper, we list the key properties of activation functions and propose a new activation function. We performed experiments on the new activation along with some of the most popular activation functions using MNIST, SVHN, CIFAR-10 and CIFAR-100 as datasets. On simple scenarios like 2 layers network MNIST classification, the choice of activation function does not have a significant impact on classification accuracy. However, when the problem difficulty is increased and the network model becomes more complex (ResNet and WRN), the choice of activation function becomes important. The new activation function was found to comparable or better performance compared to Swish. However, the activation functions cannot beat ReLU and Leaky ReLU by any significant margin. On average, throughout the experiments conducted in this study, ReLU and Leaky ReLU have the best performance.

## 6 Limitations and Future Works

Due to the limited amount of time and computational resources available for this project, we focused on relatively small datasets (less than 200MB). Performing the same tests on large datasets like ImageNet [16] would give more promising results. Most of the existing evaluations focus on deep networks. For example, in the Swish paper [21] the authors use ResNet-164 and WRN-28-10 and obtained much higher accuracy on the CIFAR-10 and CIFAR-100 datasets. We did not use those deep networks because of the time constraint. However, this distinguishes our work with the previous works [3, 21, 26].

We used two most popular residual network models: ResNet [8] and WRN [27], however there are models available like GoogleNet [24], VGGNet [23] and DenseNet [10] that we did not try. DenseNet [10] has better performance compareed to WRN and ResNet in the experiments [21, 10].

This study only provides ideal properties of an activation function and experimental analysis of a few activations. We did not provide theoretical explanations as to why certain function activation function perform better than others.

We did not try all of the activation functions like maxout [6], Swish-beta [21], Parametric ReLU [7] and SELUs [14]. These activation functions might outperform the baseline ReLU in some cases. However, based on the experiments done in the papers [21, 26] we do not expect them to have a huge improvement over the baseline.

Initialization plays an important role in the performance of neural networks [5]. In this paper we only used a small, fixed set of learning rates and weights for each model. Different initialization strategies could affect the performance of neural networks using different activation function [3, 21]. For future work, we would like to observe the impact of different hyper-parameters and initializations on the performance of activation functions.

## References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[3] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ArXiv e-prints*, November 2015.

[4] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[6] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[10] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[12] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[13] Jiqizhixin. jiqizhixin/ml-tutorial-experiment, Nov 2017.

[14] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-Normalizing Neural Networks. *ArXiv e-prints*, June 2017.

[15] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models.

[18] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[19] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning.

[20] Adam Paszke, Soumith Chintala, Ronan Collobert, Koray Kavukcuoglu, Clement Farabet, Samy Bengio, Iain Melvin, Jason Weston, and Johnny Mariethoz. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017.

[21] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions. *ArXiv e-prints*, October 2017.

[22] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[23] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*, September 2014.

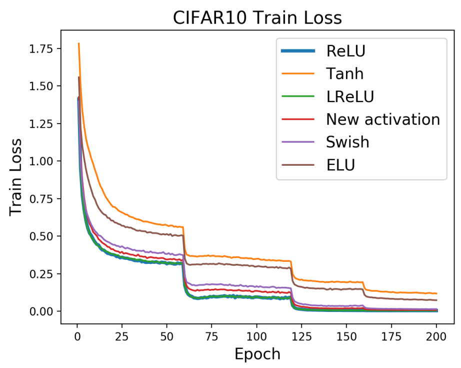
[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *ArXiv e-prints*, September 2014.

[25] Thomalm. thomalm/svhn-multi-digit, May 2017.

[26] B. Xu, N. Wang, T. Chen, and M. Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *ArXiv e-prints*, May 2015.

[27] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

## 7 Appendix



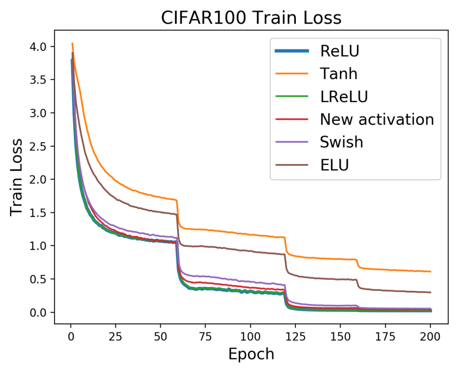


Figure 10: WRN Training Accuracy CIFAR10 and CIFAR100 batch size=128

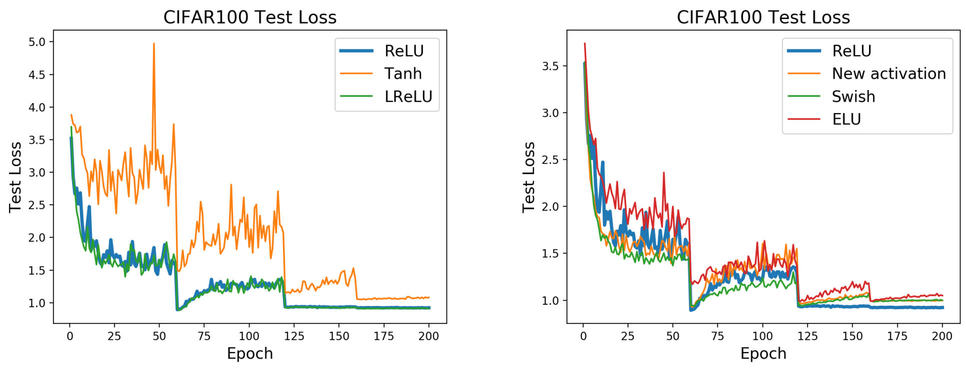


Figure 11: WRN Test Loss CIFAR10 batch size=128

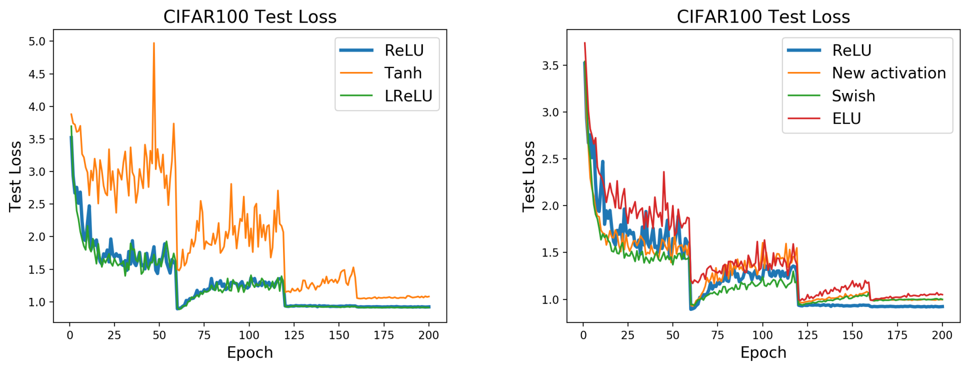


Figure 12: WRN Test Loss CIFAR100 batch size=128