E-swish: A New Novel Activation Function

Eric Alcaide

Independent Researcher, Barcelona, Spain.

ericalcaide1@gmail.com

ABSTRACT

Activation functions have a notorious impact in neural networks on both training and testing the models against the desired problem. Currently, the most used activation function is the Rectified Linear Unit (ReLU). Although some alternatives have been proposed, none of them have managed to replace ReLU as the default activation due to inconstant improvements. This paper introduces a new and novel activation function, closely related with the new activation *Swish = x*sigmoid(x) (Ramachandran et al., 2017)* which we call *E-swish.*

E-swish is just a *Swish* activation function mirrored across the identity for all positive values. We show that *E-swish* outperforms many other well-known activations on a variety of tasks and it also leads to a faster convergence.

The code to reproduce all our experiments can be found at https://github.com/EricAlcaide/E-swish

1. INTRODUCTION

The election of the activation function has a notorious impact on both training and testing dynamics of a Neural Network. A correct choice of activation function can speed up the learning phase and it can also lead to a better convergence, resulting in an improvement on metrics and/or benchmarks.

Initially, since the first neural networks where shallow, *sigmoid* or *tanh* nonlinearities were used as activations. The problem came when the depth of the networks started to increase and it became more difficult to train deeper networks with these functions (*Glorot and Bengio, 2010*).

The introduction of the *Rectifier Linear Unit (ReLU) (Hahnloser et al., 2000; Jarrett et al., 2009; Nair & Hinton, 2010)*, allowed the training of deeper networks while also providing improvements which allowed the accomplishment of new *State-of-the-Art* results *(Krizhevsky et al., 2012).*

Since then, a variety of activations have been proposed *(Maas et al., 2013; Clevert et al., 2015; He et al., 2015; Klambauer et al., 2017).* However, none of them have managed to replace *Relu* as the default activation for the majority of models due to inconstant gains and computational complexity (*Relu is simply max(0, x)*).

In this paper, we introduce a new activation function closely related to the recently proposed *Swish* function *(Ramachandran et al., 2017),* which we call *E-swish. E-swish* is just a *Swish* activation function *(x*sigmoid(x))* mirrored across the identity for all positive values. Therefore, *E-swish* can be formally described as:

$$f(x) = \begin{Bmatrix} x * \left(2 - sigmoid(x)\right) & if\ x > 0 \\ x * sigmoid(x) & if\ x \leq 0 \end{Bmatrix}$$

Our experiments show that *E-swish* systematically outperforms any other well-known activation function, providing not only a better overall accuracy, but also a faster convergence than *Relu* and *Swish*, sometimes matching the speed of *Elu*. Our experiments show that *E-swish* outperforms both *Relu* and *Swish* even when the hyperparameters are designed for *Relu*. For example, in the Wide ResNet 16-4 *(Zagoruyko & Komodakis, 2016), E-swish* provided an improvement of 0.3% better accuracy, and we expect that this difference between *Relu* and *E-swish* will increase when using deeper models.

2. E-SWISH

We propose a new activation function, which we call *E-swish*:

$$f(x) = \begin{Bmatrix} x * \left(2 - sigmoid(x)\right) & if\ x > 0 \\ x * sigmoid(x) & if\ x \leq 0 \end{Bmatrix}$$

Where $sigmoid(x) = 1/(1 + \exp(-x))$.

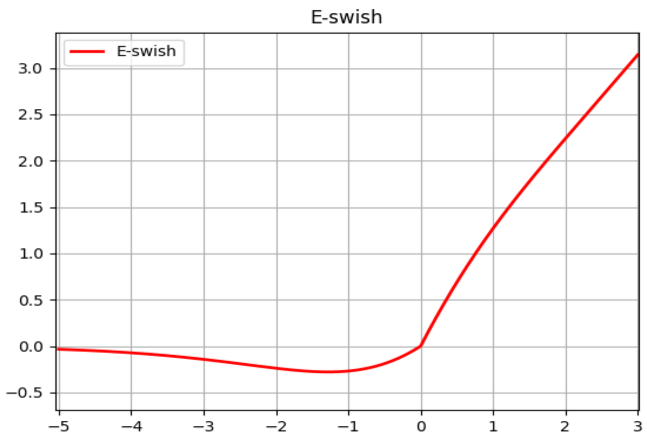

Figure 1. E-swish activation function.

Like both *Relu* and *Swish*, *E-swish* is unbounded above and bounded below. *E-swish* is also nonsmooth, *like Relu* and unlike *Swish.* However, like *Swish* and unlike *Relu*, *E-swish* is nonmonotonic. The property of non-monotonicity is almost exclusive of *Swish* and *E-swish.*

Another exclusive feature of both, *Swish* and *E-swish,* is that there is a region where the derivative is greater than 1, reaching 1.5 at its highest point.

The derivative of *Swish* is:

$$
\begin{aligned}
f'(x) = {} & \sigma(\mathrm{x}) + \mathrm{x} \cdot \sigma(\mathrm{x})\big(1 - \sigma(\mathrm{x})\big) = \\
& = \sigma(\mathrm{x}) + \mathrm{x} \cdot \sigma(\mathrm{x}) - \mathrm{x} \cdot \sigma(\mathrm{x})2 = \\
& = \mathrm{x} \cdot \sigma(\mathrm{x}) + \sigma(\mathrm{x})\big(1 - \mathrm{x} \cdot \sigma(\mathrm{x})\big) = \\
& = \mathrm{f(x)} + \sigma(\mathrm{x})\big(1 - \mathrm{f(x)}\big)
\end{aligned}
$$

Where $\sigma(\mathrm{x}) = sigmoid(x)$, described before.

The derivative of *E-swish,* therefore, is:

$$
f'(x) = \begin{cases} 2 - \big(\mathrm{f(x)} + \sigma(\mathrm{x})\big(1 - \mathrm{f(x)}\big)\big) & if\ x > 0 \\ \mathrm{f(x)} + \sigma(\mathrm{x})\big(1 - \mathrm{f(x)}\big) & if\ x \leq 0 \end{cases}
$$

As it can be inferred from these formulas, the *Figure 1,* and as it can be seen in *Figure 2,* the derivative of *E-swish* is bigger than 1 in a small region. For the positive part, it goes down until 0.9 and then approaches to 1. For the negative part, the derivative of *E-swish* is the same as *Swish.* This may be confusing, but it provides a faster learning and we show experimentally that both *E-swish* and *Swish* are able to train deeper networks than *Relu* when using Batch Normalization *(Ioffe & Szegedy, 2015).*

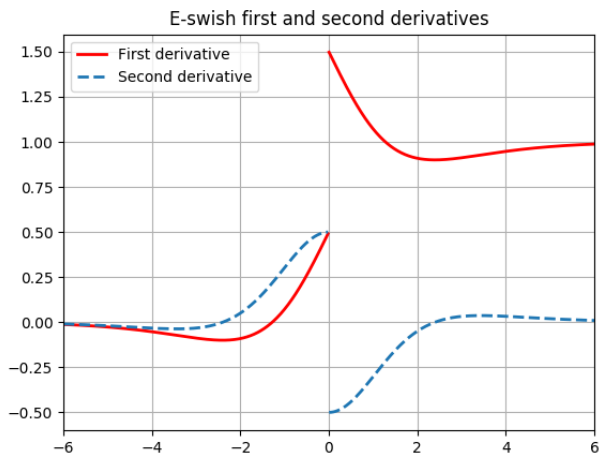

Figure 2. First and second derivatives of *E-swish.*

E-swish can be implemented as a custom activation in some popular deep learning libraries (eg. `K.maximum(x*K.sigmoid(x), x*(2-K.sigmoid(x)))` when using *Keras* or `tf.maximum(x*tf.sigmoid(x), x*(2-tf.sigmoid(x)))` when using *Tensorflow…*). Implementations of *E-swish* in some widely used deep learning frameworks will be provided together with the code to reproduce the experiments performed in this paper.

2.1. PROPERTIES OF E-SWISH

This paper takes as a reference the work of the original *Swish* paper (*Ramachandran et al., 2017*). For that reason, we compare *E-swish* to *Relu* and *Swish* as we consider them the basis to work on and compare to. We don't provide extensive comparisons with other activation functions, since they're provided in the original *Swish* paper.

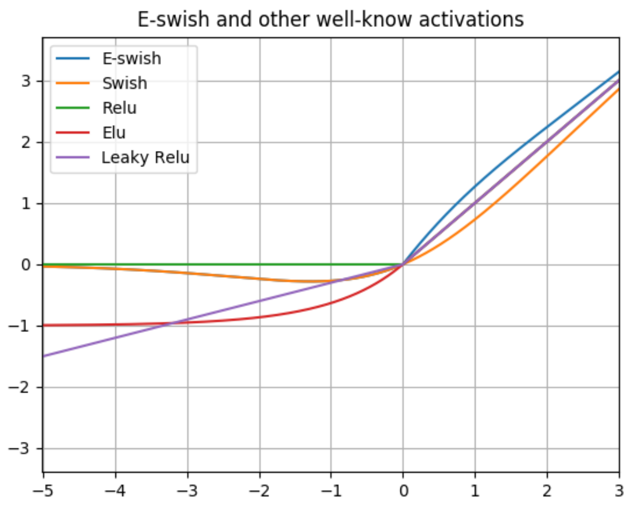

Figure 3. E-swish and other well-known activation functions. Best viewed in color.

As it can be seen in *Figure 3, E-swish* (blue) is the reflection of *Swish* (orange) across the identity for all positive values (*Relu/Leaky Relu for x>0*).

It's very difficult to determine why some functions perform better than others given the presence of a lot of confounding factors. Despite of this, we believe that the non-monotonicity of *E-swish* favours its performance. The fact that the gradients for the negative part of the function approach zero can also be observed in *Swish, Relu* and *Softplus* activations. However, we believe that the particular shape of the curve described in the negative part, which gives both *Swish* and *E-swish* the non-monotonicity property, improves performance since they can output small negative numbers, unlike *Relu* and *Softplus.*

Inspired by the original *Swish* paper, we plot below the output landscape of a random network, which is the result of passing a grid in form of points coordinates. The network is composed of 6 layers with 128 neurons each and initialized to random values.

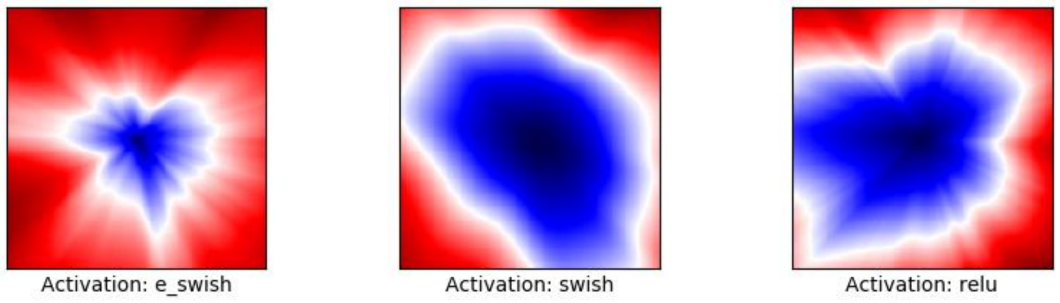

Figure 4. Output landscape for a random network with different activation functions. Best viewed in color.

In the original *Swish* paper, the improvements associated with the *Swish* function were supposed to be caused by the smoothness of the *Swish* output landscape. As it's true that smoothness plays a major role in optimization, we can, however, see a great region both in *Relu* and *Swish* output landscapes where the values are almost equal. In constrast, it doesn't happen with the *E-swish* activation, where there's a very small region in the centre where the output value is minimum. This can be seen as a more strict constraint that *E-swish* applies to the network.

If we plot the output landscapes in 3D, as can be seen in *Figure 5,* it can be observed that the slope of the *E-swish* landscape is higher than the one in *Relu* and *Swish.* Therefore, we can infer that, probably, *E-swish* will show a faster learning than *Relu* and *Swish.* As a reference, we also plot the landscape of the *Elu* activation function since it also provides a faster learning than *Relu,* and its slope is higher too.

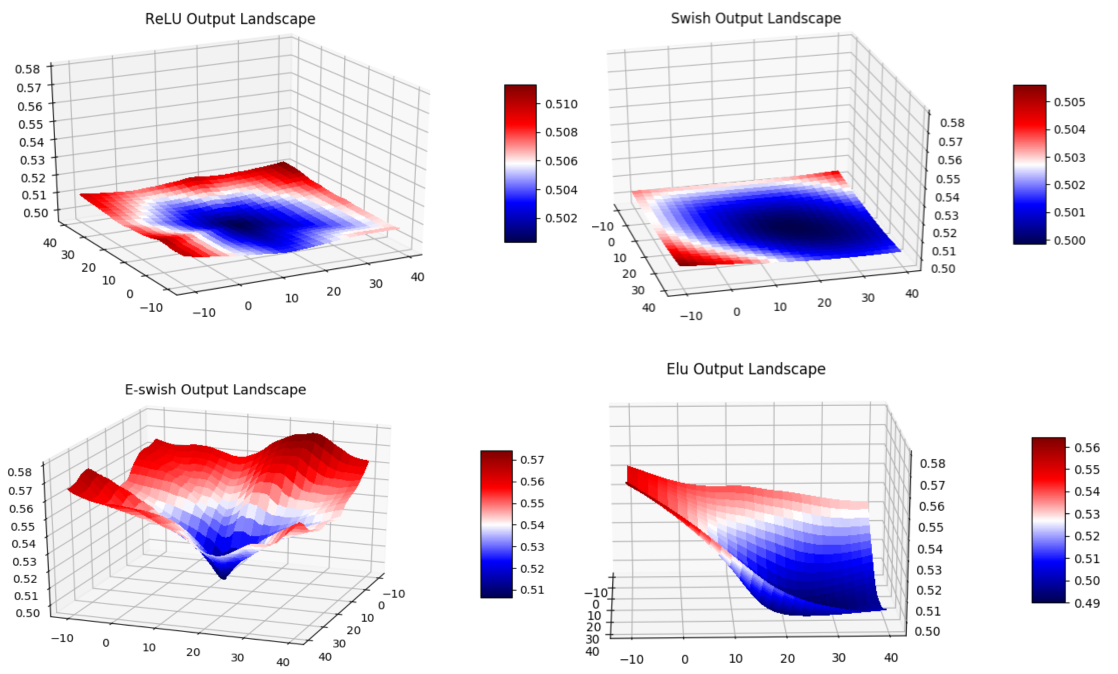

Figure 5. 3D projection of the output landscape of a random network. Values were obtained by fitting the coordinates of each point in a grid to a 6-layer, with 128 neurons each, randomly initialized neural network. Best viewed in color.

Figures 4 and *5* are very similar to the one presented in *Ramachandran et al., 2017.*

Given the shape of the *E-swish* output landscape, we hypothesize that *E-swish* would benefit from a decaying learning rate as it approaches convergence. This hypothesis is proved in our experiments since *E-swish* increases the performance when the learning rate becomes smaller as it approaches the convergence point.

3. EXPERIMENTS

All our experiments were run either on a Nvidia GeForce GT 730 (2gb of memory) or a Nvidia Tesla K80 (12gb of memory).

3.1. TRAINING DEEP NETWORKS

First of all, we want to prove that *E-swish* has also the ability to train deeper networks than *Relu*, shown for *Swish* in *Ramachandran et al., 2017*. We conduct a similiar experiment to the one performed in the paper mentioned before.

We train fully connected networks of different depths on MNIST with 512 neurons each layer. We use SGD as optimizer, the initial learning rate is set to 0.01 and momentum to 0.9. We multiply the learning rate by 0.35 whenever there are 2 epochs with no improvement in the validation accuracy. We train each network for 15 epochs although we terminate the training earlier in case that there is no improvement in the validation accuracy for 5 epochs. We use Glorot uniform initialization *(Glorot & Bengio, 2010)* and a batch size of 128. No dropout is applied.

We don't use residual connections, since they would allow to train deeper networks. We use Batch Normalization when the last layer's index (starting at 0) satisfies that *i%3=1* where *i* is the index and % is the modulus operator. Our experiment shows that *E-swish* performs very similar to *Swish* in very deep networks.

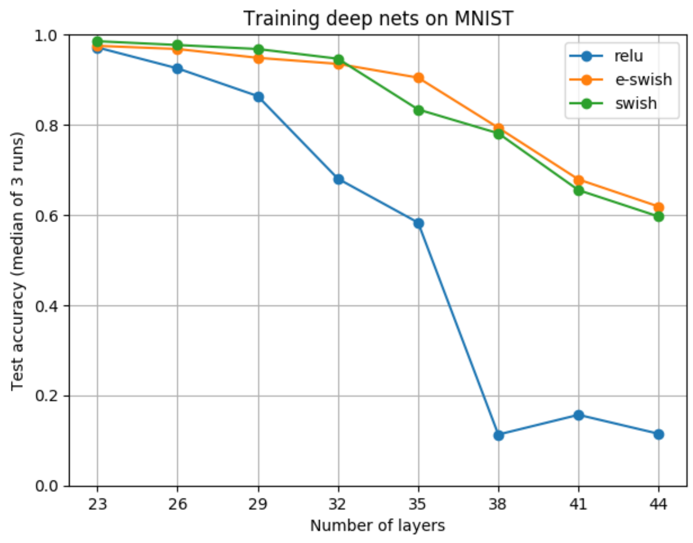

Figure 6. Test accuracy (median of 3 runs) on MNIST for fully connected networks with different activations which depths go from 23 to 44 layers.

In our experiments, the three activations (*Relu*, *Swish* and *E-swish*) perform almost equally up to 23 layers. Then, we can see a clear divergence between *Re*lu and *Swish / E-swish*, where both *Swish* and *E-swish* outperform *Relu* by a large margin in the interval of 23-50 layers, as seen in the *Figure 6* above.

This experiment proves that both *Swish* and *E-swish* outperform *Relu* when training very deep networks since they achieve better test accuracies.

3.2. BENCHMARKING E-SWISH AGAINST OTHER WELL-KNOWN ACTIVATIONS

We benchmarked *E-swish* against the following activation functions:

- Relu: $ReLU(x) = \max(0, x)$
- Swish: (*Ramachandran et al., 2017.*) $Swish(x) = x * sigmoid(x)$
- Elu: *(Clevert et al., 2015)* $Elu(x) = \begin{cases} x & if\ x \geq 0 \\ \exp(x) - 1 & if\ x < 0 \end{cases}$
- Leaky Relu: *(Maas et al., 2013)* $LReLU(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha x & if\ x < 0 \end{cases}$

First, we conduct some small experiments to show the improvements provided by *E-swish* on a range of models and architectures.

Our experiments showed that *E-swish* consistently outperforms any other activation, while *Swish* showed inconsistent improvements against *Relu*. We also noticed that *E-swish* provides a faster learning, matching sometimes the speed of *Elu*, and converging to a better optima.

We conducted our experiments on both single model performance and a median of 3 runs. In this section we will provide an extensive explanation of our experiments.

3.2.1. MNIST

For MNIST, we first compare the different activations on a single model performance. We train a fully connected network of 5 layers with the following number of neurons: 200, 100, 60, 30, 10, with dropout of 0.2. We use SGD with learning rate set to 0.1 and no momentum, we also use Glorot uniform initialization *(Glorot & Bengio, 2010)* and a batch size of 64. We train each network for 20 epochs.

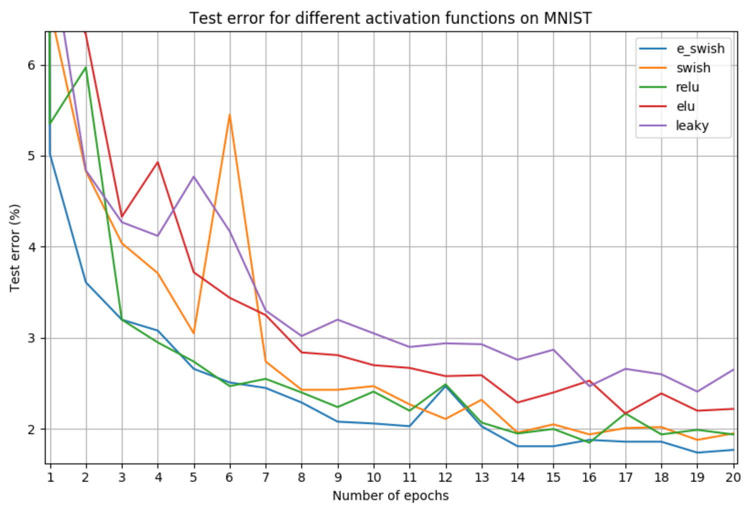

Figure 7. Training curves for MNIST with 20 epochs.

As it can be seen in *Figure 7, E-swish* provides a faster learning and a better overall accuracy on a single model performance.

Since the better performance exhibited before can be derived from a better random initialization, we perform the experiment again with the same implementation details but we compute the median of 3 runs. We obtain the following results:

Activation function	**Accuracy (%)**
E-swish	98.46
Swish	98.31
Relu	98.30
Elu	97.92
LeakyRelu (alpha=0.3)	97.71

Table 1. Accuracy on MNIST for a fully connected network of 5 layers. Results were computed over a median of 3 runs.

As can be seen in *Table 1*, *E-swish* provided an improvement of 0.16% against *Relu*, 0.15% against *Swish* and a bigger difference against *Elu* and *Leaky Relu*. This experiment highlights the superiority of *E-swish* against *Relu*, *Swish*, *Elu* and *Leaky Relu* in fully connected networks, even when they are shallow. We used the same learning rate for all the activations, although we believe that given the properties of *E-swish*, it would benefit more than the others from a decaying learning rate.

3.2.2. CIFAR-10

The CIFAR-10 dataset *(Krizhevsky & Hinton, 2009)* consists of 60k colored images with 32x32 pixels each of 10 different classes. 50k training images are used for training and 10k for testing purposes.

For this dataset, we measure the performance of *E-swish* relative to other activations.

- ***SIMPLE CNN***

First, we compare the different activations on a simple CNN (Convolutional Neural Network) to show the difference on the performance for the different activations. The CNN achieves ˜80% accuracy after 15 epochs, although we believe it can achieve ±85% on a median of 5 runs and further training. This experiment is not intended to achieve state of the art results, but to provide a small example to prove that *E-swish* performs better than the other commonly used activation functions on a CNN architecture.

Our simple CNN has the following structure:

Conv - MaxPool - Dropout - Conv - MaxPool - Dropout - Conv - Dropout - Conv - MaxPool - Dropout - Conv - MaxPool – Dropout - FC - FC.

The number of neurons/filters goes as follows: 32, 64, 64, 128, 128, 64, 10 respectively. We use Max Pooling of 2 with stride 2. Dropout rate is set to 0.15. We use Glorot uniform initialization *(Glorot & Bengio, 2010)* and the Adamax optimizer *(Kingma & Ba, 2015)* with learning rate set to 0.002, β_1=0.9, β_2=0.999, ε=1e-08 and schedule decay of 0.004. We train each network for 15 epochs and no data augmentation techniques are used.

We compute the median of 3 runs for each activation function and we get the following test accuracy rates:

Activation function	**Accuracy (%)**
E-swish	83.35
Swish	81.88
Relu	82.06
Elu	81.81
LeakyRelu (alpha=0.3)	82.22

Table 2. Accuracy on Cifar10 for a simple CNN. Results were computed over a median of 3 runs.

As can be seen in *Table 2*, *E-swish* outperforms any other well-known and widely-used activation function in this simple CNN.

- ***DEEPER CNN***

We then compare *E-swish* against the other activations on a deeper CNN, which we believe can achieve >90% accuracy on Cifar10. This CNN is designed to achieve a good tradeoff between the number of parameters and accuracy, since it has only 330K parameters. Its structure is the following:

Conv - BatchNorm - Conv - BatchNorm - MaxPool - Dropout –

Conv - BatchNorm - Conv - BatchNorm - MaxPool - Dropout –

Conv - BatchNorm - Conv - BatchNorm - MaxPool - Dropout - FC

The number of filters is, in order, 32, 32, 64, 64, 128, 128, 10. Max pooling of 2 with stride 2 is used. We use weight decay of 0.0001 and dropout rate is set to 0.2, 0.3 and 0.4 respectively. We use Glorot uniform initialization *(Glorot & Bengio, 2010)* and the RMSprop optimizer (Tieleman & Hinton, 2012) with initial learning rate of 0.001, multiplying the learning rate approximately by 0.5 at 75 epochs for the first time, and then every 25 epochs. Rho (ρ) is set to 0.9, epsilon (ε) to 10^{-8} and decay rate is set to 0. We train each network for 250 epochs. We use a batch size of 128, *mean/std* preprocessing and the following data augmentation techniques: horizontal flipping, random cropping of 42x42px images padded with the nearest pixel value and random rotations of up to 15 degrees.

Due to computational constraints, we were only able to benchmark *E-swish* against *Relu* and *Swish* on a median of 3 runs. We also tested *Leaky Relu* and *Elu* on a single model performance. The results we obtained will be provided below.

Activation function	**Test error (%)**
E-swish	7.61
Swish	8.27
Relu	8.13
Elu	8.95*
LeakyRelu (alpha=0.3)	9.25*

Table 3. Test error rates for Cifar10 after 250 epochs of training. Results with an asterisk (*) were computed on a single model performance.

As *Table 3* shows*, E-swish* outperforms any other activation by, at least, 0.5%. Considering that hyperparameters were designed for *Relu*, each network had the same num. of parameters, same conditions of training, and that there is no real improvement since the 200th epoch, it is really significative. <u>*E-swish* converges to a better optima.</u>

We suppose that an improvement of up to 1% (0.5-1%) can be achieved on a median of 3 runs for *Elu* and *Leaky Relu* based on improvements observed for *E-swish, Swish and Relu*. However, *E-swish* still performs better than the other activations.

The results obtained by *E-swish* are comparable to or outperform the ones obtained in *Network in Network (Lin et al., 2013), Deeply Supervised Net (Lee et al., 2015), Highway Network (Srivastava et al., 2015),* and *Fitnets: Hints for thin deep nets (Romero et al., 2015),* <u>while requiring much fewer parameters.</u>

We believe that a further improvement in a range from 0.15% to 0.5% can be achieved by computing the median of 5 runs. However, we don't take this into account when doing comparisons with other models.

- ***WIDE RESIDUAL NETWORK 16-4***

Finally, we compare *E-swish* against *Relu* on a Wide Residual Network *model (Zagoruyko & Komodakis, 2016).* We've chosen the WRN 16-4 with no dropout since we were not able to train bigger models due to computational constraints. We use the exact implementation provided by the authors of the original paper, replacing the original *Relu* by the *E-swish* activation function.

E-swish presented an improvement in performance when compared to *Relu*, <u>with 4.71% error rate on a median of 5 runs, while the original paper states 5.02% test error rate</u> for the WRN 16-4 model on a median of 5 runs. This means a 0.3% improvement. Since the *E-swish* activation is more complex than *Relu* and, therefore, more computationally expensive, we expected that more time to train the models would be required. However, we noticed that *E-swish* balanced this drawback by providing better results.

For example, *E-swish* reached an accuracy of 94.96% on a median of 3 runs, which means a 5.04% error rate, while *Relu* needs a median of 5 runs to achieve similar results.

<u>*E-swish* achieves an accuracy of 95.29%</u> over a median of 5 runs while <u>*Relu* achieves 94.98% of accuracy under the same exact conditions.</u>

4. *FURTHER WORK*

Further experiments couldn't be performed due to the resource-intensive nature of training vey deep networks and computational constraints by the time of the publication of this paper. For that reason, we would like to invite everyone to perform their own experiments with *E-swish* in order to get a more extensive proof of its superiority against other activations.

Especially, we are interested in testing *E-swish* on Wide Residual Networks, DenseNets (Huang et al., 2016), ResNets (He et al., 2015), GANs (Goodfellow et al., 2014), Autoencoders, RNNs, NASNets (Zoph et al., 2017), and any other model that achieves state of the art results in order to see if it achieves better results in these models.

Further experimentation and testing will be performed by the authors of this paper and the results will be updated as soon as possible.

We are also interested in the possible application of *E-swish to* SNNs *(Liu et al., 2017).*

5. *CONCLUSION*

In this paper, we have presented a new novel activation function, which we call *E-swish*, which can be formally described as:

$$f(x) = \begin{Bmatrix} x * \left(2 - sigmoid(x)\right) & if\ x > 0 \\ x * sigmoid(x) & if\ x \leq 0 \end{Bmatrix}$$

Our experiments used models with hyperparameters that were designed for *Relu* and just replaced *Relu* by *Swish, E-swish* or another activation function. Although this was a suboptimal technique, *E-swish* outperformed both *Swish* and *Relu* on a variety of problems. *E-swish* could also benefit from hyperparameters designed with its properties in mind, specially from a decaying learning rate.

Although we have been unable to achieve State-of-the-Art results due to the resource-intensive nature of training huge models, we've shown that *E-swish* consistently outperforms any other activation function on a range of different problems, depths and architectures and we have managed to improve an existing result for a relatively big and recent model by changing the original activation (*Relu*) by *E-swish.*

E-swish is a more complex activation than *Swish* or *ReLU* and therefore, it is slower to train for the same number of epochs. However, *E-swish* often balances this drawback by providing a faster and/or better learning.

Further work could apply this new and novel function to more complex models and produce new *State-of-the-Art* results for different datasets.

REFERENCES

- [1] Xavier Glorot and Yoshua Bengio: *Understanding the difficulty of training deep feedforward neural networks. 2010.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.207.2059&rep=rep1&type=pdf
- [2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. *Generative Adversarial Networks.* 2014. https://arxiv.org/pdf/1406.2661.pdf
- [3] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. 2000. https://www.ncbi.nlm.nih.gov/pubmed/10879535*
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition.* 2015. https://arxiv.org/pdf/1512.03385.pdf
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015. https://arxiv.org/pdf/1502.01852.pdf*
- [6] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. *Densely Connected Convolutional Networks.* 2016. *https://arxiv.org/pdf/1608.06993.pdf*
- [7] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. https://arxiv.org/pdf/1502.03167.pdf
- [8] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. *What is the best multi-stage architecture for object recognition?* 2009. http://yann.lecun.com/exdb/publis/pdf/jarrett-iccv-09.pdf
- Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization. 2014. https://arxiv.org/pdf/1412.6980.pdf*
- [9] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. *Self-normalizing neural networks.* https://arxiv.org/pdf/1706.02515.pdf
- [10] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images.* Technical report, Technical report, University of Toronto, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks.* 2012. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- [12] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang and Zhuowen Tu. *Deeply-Supervised Nets*. 2015. https://arxiv.org/pdf/1409.5185.pdf
- [13] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network. 2013. https://arxiv.org/pdf/1312.4400.pdf*
- [14] Qian Liu and Steve Furber. *Noisy Softplus: A Biology Inspired Activation Function.* 2016. https://link.springer.com/chapter/10.1007/978-3-319-46681-1_49
- [15] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. *Rectifier nonlinearities improve neural network acoustic models.* 2013. https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf
- [16] Vinod Nair and Geoffrey E Hinton. *Rectified linear units improve restricted boltzmann machines.* 2010. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.6419&rep=rep1&type=pdf
- [17] Prajit Ramachandran, Barret Zoph, Quoc V. Le: *Swish: a Self-Gated activation function. 2017.* https://arxiv.org/pdf/1710.05941v1.pdf
- [18] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. *FitNets: Hints for Thin Deep Nets.* 2014. https://arxiv.org/pdf/1412.6550.pdf
- [19] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. *Highway Networks*. 2015. https://arxiv.org/pdf/1505.00387.pdf
- [19] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report, 2012.
- [20] Sergey Zagoruyko and Nikos Komodakis. *Wide residual networks*. 2016. https://arxiv.org/pdf/1605.07146.pdf
- [21] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. *Learning Transferable Architectures for Scalable Image Recognition.* 2017. https://arxiv.org/pdf/1707.07012.pdf