Mish: A Self Regularized Non-Monotonic Neural Activation Function

Diganta Misra
mishradiganta91@gmail.com

Abstract

The concept of non-linearity in a Neural Network is introduced by an activation function that serves an integral role in the training and performance evaluation of the network. Over the years of theoretical research, many activation functions have been proposed, however, only a few are widely used in mostly all applications which include ReLU (Rectified Linear Unit), TanH (Tan Hyperbolic), Sigmoid, Leaky ReLU and Swish. In this work, a novel activation function, Mish is proposed which can be defined as: $f(x) = x \cdot tanh(softplus(x))$. *The experiments show that Mish tends to work better than both ReLU and Swish along with other standard activation functions in many deep networks across challenging datasets. For instance, in Squeeze Excite Net- 18 for CIFAR 100 classification, the network with Mish had an increase in Top-1 test accuracy by 0.494% and 1.671% as compared to the same network with Swish and ReLU respectively. The similarity to Swish along with providing a boost in performance and its simplicity in implementation makes it easier for researchers and developers to use Mish in their Neural Network Models.*

1. Introduction

The mathematical computation in every deep neural network model includes a linear transformation followed by an activation function. This activation function is the key to introducing non-linearity in the network. Activation functions play a crucial role in the performance of every deep network. Currently, in the deep learning community, two activation functions have been predominately being used as the standard for all applications. These two are: Rectified Linear Unit (ReLU) [1,2,3] which can be defined by $f(x) = \max(0, x)$ and Swish [4,5] which can be defined as: $f(x) = x \cdot sigmoid(x)$.

ReLU has been used as the standard/ default activation function in mostly all applications courtesy to its simple implementation and consistent performance as compared to other activation functions. Over the years, many activation functions have been proposed to replace ReLU which includes Square Non-Linearity (SQNL) [6], Exponential Linear Unit (ELU), Parametric Rectified Linear Unit (PReLU) [7] along with many others. However, the simplicity and efficiency of ReLU remained unchallenged throughout, until Swish Activation Function was released which showcased strong and improved results on many challenging benchmarks. Unlike ReLU, Swish is a smooth non-monotonic activation function and similar to ReLU, it is bounded below and unbounded above. Swish demonstrated significant improvements in top-1 test accuracy across many deep networks in challenging datasets like ImageNet.

In this paper, *Mish*, a novel neural activation function is introduced. Similar to Swish, Mish is a smooth and non-monotonic activation function which can be defined as: $f(x) = x \cdot tanh(softplus(x)) = x \cdot tanh(\ln(1 + e^x))$ Throughout the extensive testing and experimentation conducted Mish demonstrated better results than both Swish and ReLU. For example, during the classification of CIFAR-100 dataset using a Squeeze Excite -18 Network [8] with Mish resulted in an increase in Top-1 test accuracy by 0.494% and 1.671% as compared to the same network with Swish and ReLU respectively. Mish provides near consistent improvement in accuracy over Swish and ReLU as seen in the case of CIFAR-100 classification using a MobileNet v2 [9] where the network with Mish had an increase in Top-1 test accuracy by 1.385% over Swish and 0.8702% over ReLU.

2. Mish

Mish is a novel smooth and non-monotonic neural activation function which can be defined as:

$$f(x) = x \cdot tanh(\varsigma(x)) \tag{1}$$

where, $\varsigma(x) = \ln(1 + e^x)$ is the softplus activation [10] function. The graph of Mish is shown in Figure 1.

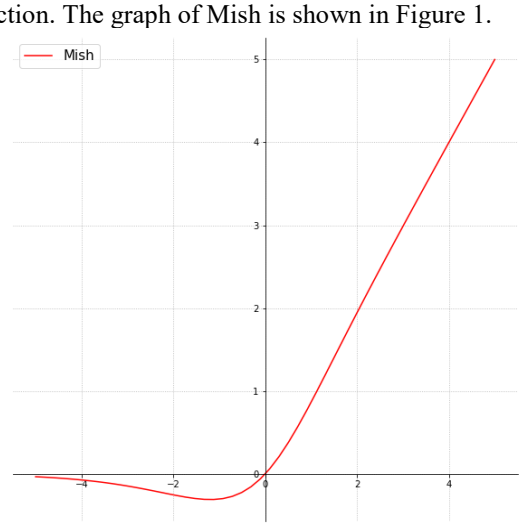

Figure 1. Mish Activation Function

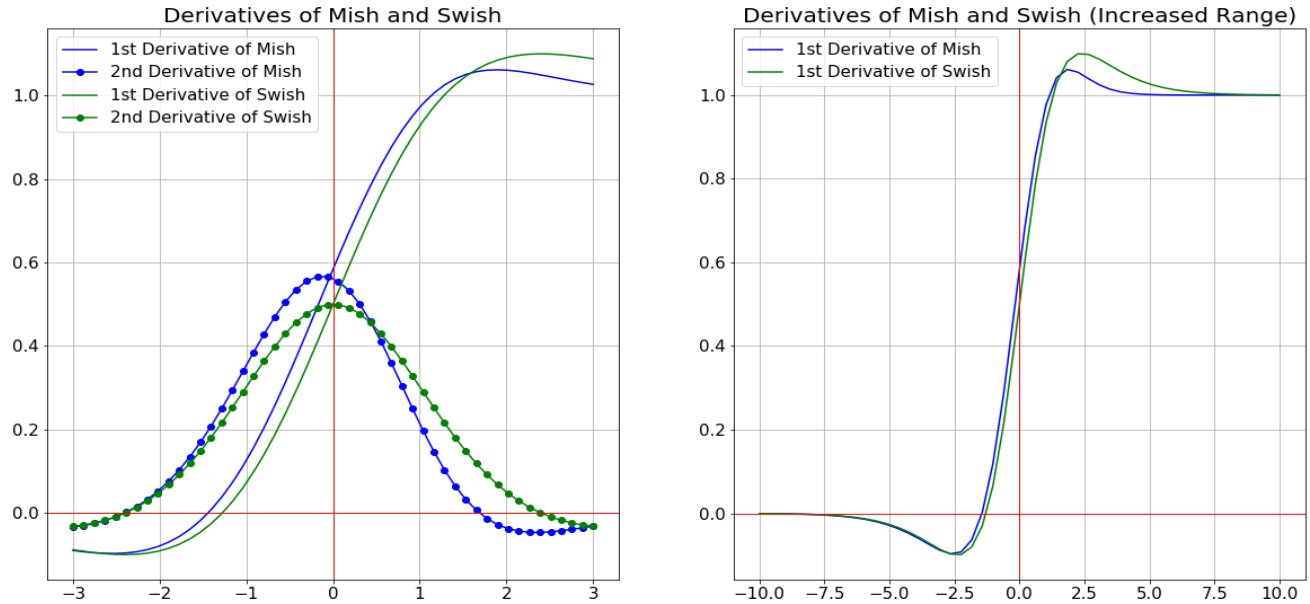

Figure 2. Comparison between 1st and 2nd derivative of Mish and Swish

Like both Swish and ReLU, Mish is bounded below and unbounded above with a range [≈-0.31, ∞). The derivative of Mish shown in Figure 2 is defined as:

$$f'(x) = \frac{e^x \omega}{\delta^2} \qquad (2)$$

where, $\omega = 4(x+1) + 4e^{2x} + e^{3x} + e^x(4x+6)$ and $\delta = 2e^x + e^{2x} + 2$. The minimum of Mish is observed to be at x≈-1.1924 with a magnitude of ≈-0.30884. Mish takes inspiration from Swish by using a property called Self Gating, where the scalar input is provided to the gate. The property of Self-gating is advantageous for replacing activation functions like ReLU (point-wise functions) which take in a single scalar input without requiring to change the network parameters.

Mish can be easily implemented using any standard deep learning framework by defining a custom activation layer. In Tensorflow [11], the function definition of Mish can be written as x * tf.math.tanh(tf.softplus(x)) while in Torch [12] it is x * torch.tanh(F.softplus(x)). For improved results over ReLU, it is advised to use a slightly lower learning rate for Mish.

2.1. Properties of Mish

Although it's difficult to explain the reason why one activation function performs better than another due to many other training factors, the properties of Mish like being unbounded above, bounded below, smooth and non-monotonic, all play a significant role in the improvement of results. Figure 3 shows the different commonly used activation functions along with the graph of Mish activation for comparison.

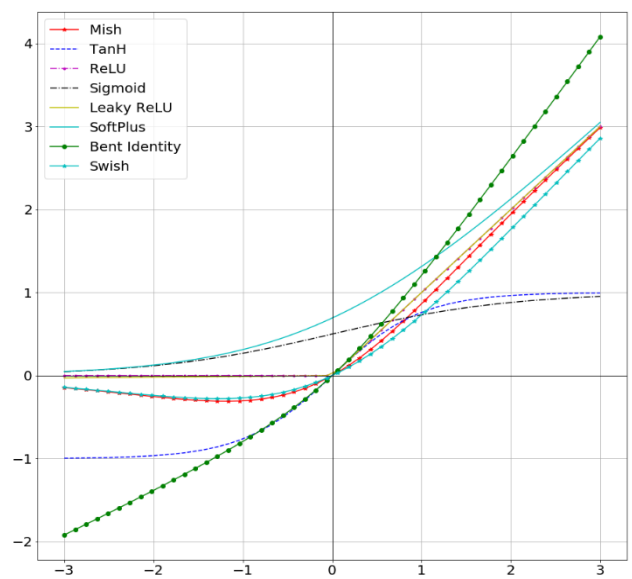

Figure 3. Common Activation Functions

Table 1 provides a detailed summary of the properties of Mish Activation Function.

Table 1. Properties Summary of Mish

Order of Continuity	C^∞
Monotonic	No
Monotonic Derivative	No
Saturated	No
Approximates Identity at Origin	Yes (Approximates half of identity at Origin)

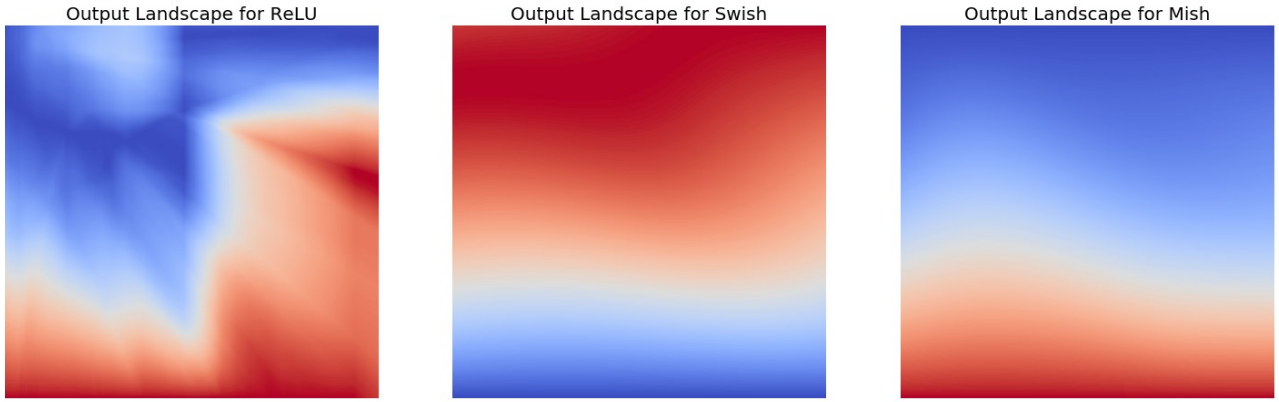

Figure 5. Output Landscape of a Random Neural Network with ReLU, Swish and Mish

Being unbounded above is a desirable property for any activation function since it avoids saturation which generally causes training to drastically slow down due to near-zero gradients [13]. Being bounded below is also advantageous since it results in strong regularization effects. The non-monotonic property of Mish causes small negative inputs to be preserved as negative outputs as shown in Figure 4, which improves expressivity and gradient flow. The order of continuity being infinite for Mish is also a benefit over ReLU since ReLU has an order of continuity as 0 which means it's not continuously differentiable causing some undesired problems in gradient-based optimization.

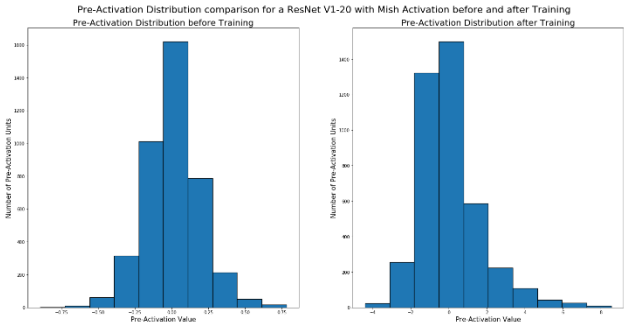

Figure 4. Pre-Activation Distribution comparison before and after training for a ResNet v1-20 with Mish.

Mish being a smooth function also plays an important role in explaining the improvement in results as it helps with effective optimization and generalization. The output landscape of 5 layer randomly initialized neural network was compared for ReLU, Swish, and Mish. The observation as shown in Figure 5, clearly depicts the sharp transition between the scalar magnitudes for the co-ordinates of ReLU as compared to Swish and Mish. Smoother transition results in smoother loss functions which are easier to optimize and hence the network generalizes better which partially explains why Mish outperforms ReLU. However, in this regard, Mish and Swish are extremely similar in their corresponding output landscapes.

3. Comparison of variation in hyper-parameters

3.1. MNIST

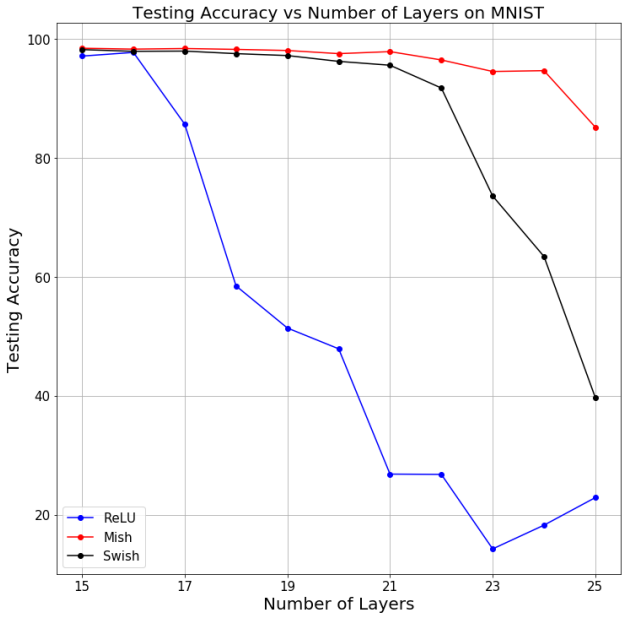

Figure 6. Testing Accuracy v/s Number of Layers on MNIST for Mish, Swish and ReLU.

To observe how increasing the number of layers in a network while maintaining other parameters constant affects the test accuracy, fully connected networks of varying depths on MNIST, with each layer having 500 neurons were trained. Residual Connections [14] were not used because they enable the training of arbitrarily deep networks. BatchNorm [15] was used to lessen the dependence on initialization along with a dropout [16] of 25%. The network is optimized using SGD [17] on a batch size of 128, and for fair comparison, the same learning

rates for each activation function was maintained. In the experiments, all 3 activations maintained nearly the same test accuracy for 15 layered Network. Increasing the number of layers from 15 gradually resulted in a sharp decrease in test accuracy for Swish and ReLU, however, Mish outperformed them both in large networks where optimization becomes difficult as shown in Figure 6.

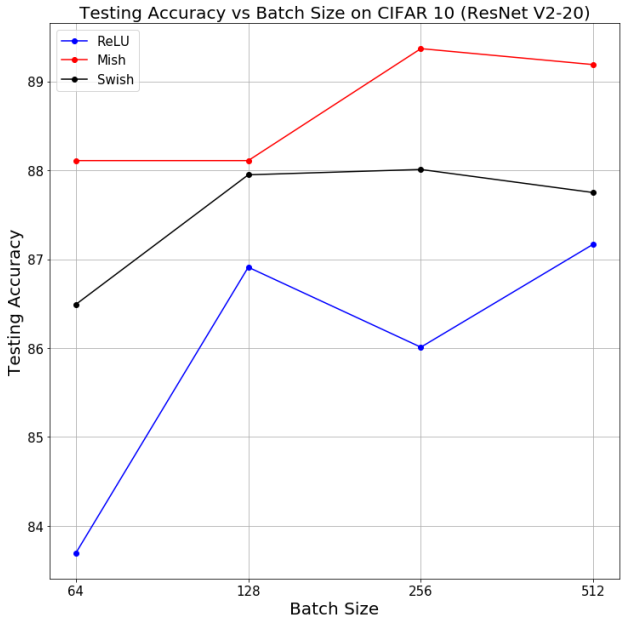

Figure 7. Test Accuracy v/s Batch Size on CIFAR-10 for Mish, Swish and ReLU.

The consistency of Mish was also observed by increasing Batch Size for ResNet v2-20 on CIFAR-10 for 50 epochs while keeping all other network parameters constant for fair comparison as demonstrated in Figure 7.

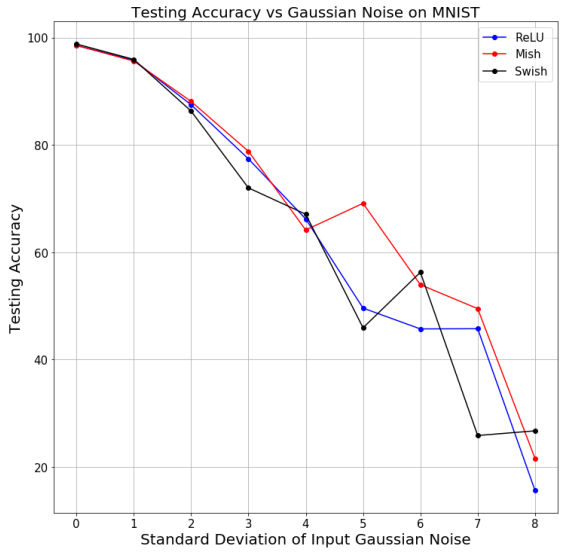

Figure 8. Test Accuracy v/s Input Gaussian Noise on MNIST for Mish, Swish and ReLU.

The robustness of Mish was evaluated in Noisy Input conditions where using different non-linearities including Swish and ReLU in a standard network, the drop in test top-1 accuracy and the increase in loss were recorded. The input MNIST data was corrupted with additive zero-centered Gaussian Noise with increasing standard deviation values to a maximum of 8. As shown in Figure 8 and Figure 9, Mish consistently had a lower loss and a higher test accuracy with varying intensity of input Gaussian Noise.

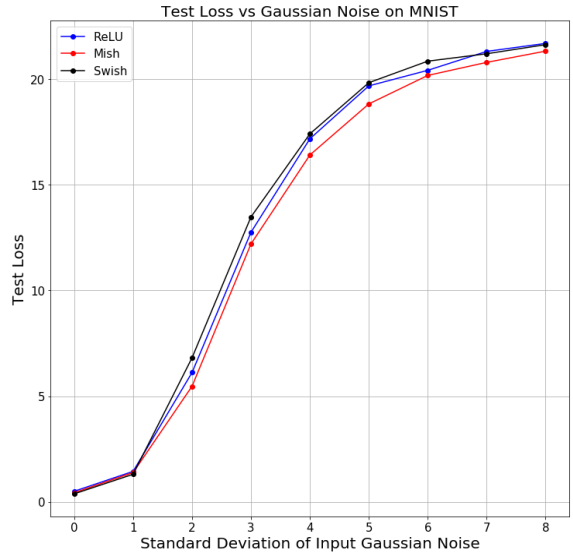

Figure 9. Test Loss v/s Input Gaussian Noise on MNIST for Mish, Swish and ReLU

The effect of using different optimizers was also experimented upon to evaluate the efficiency of Mish as compared to that of Swish as demonstrated in Figure 10.

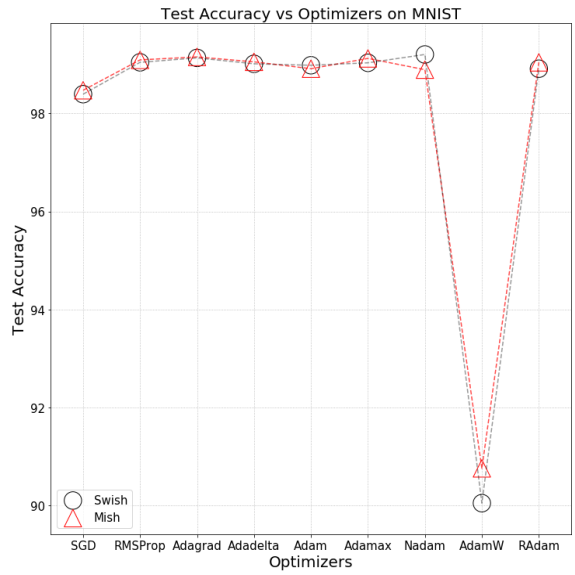

Figure 10. Test Accuracy v/s Optimizers for Mish and Swish.

Learning Rate (LR) is crucial towards optimization processes in Neural Networks and the optimal LR value is often debated upon. The results in Figure 11 where Mish was compared with Swish using different learning rate values in a simple MNIST classifier with SGD optimizer shows Mish to match or outperform Swish in most of the used LR- values from a range of [10^{-5}, 10^{-1}] with incremental steps of the value of 10. This is also demonstrated that Mish performs especially better with lower learning rate values.

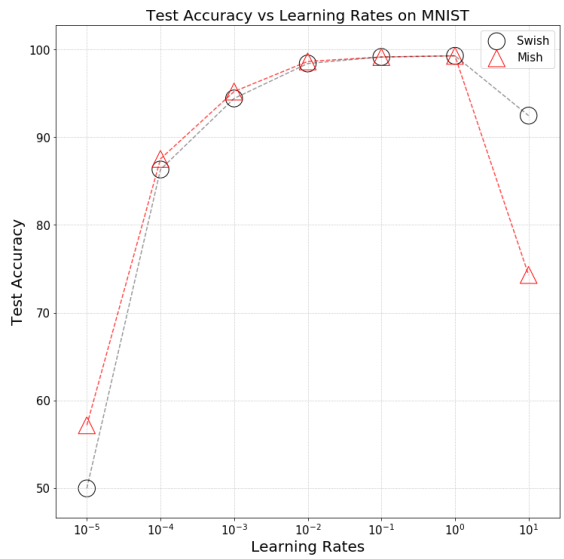

Figure 11. Test Accuracy v/s Learning Rates for Mish and Swish.

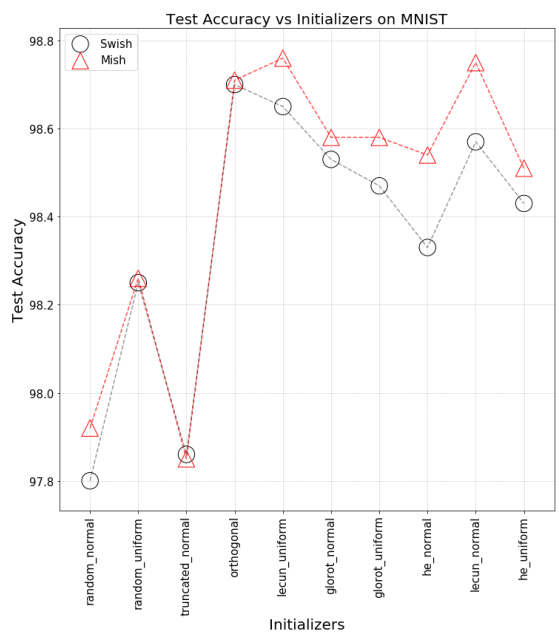

Figure 12. Test Accuracy v/s Initializers for Mish and Swish.

Figure 12 shows the comparison between Mish and Swish in terms of Test Accuracy for a simple MNIST Classifier using different initializers. Mish matched or outperformed Swish in mostly all of the initializers used in the experiment. The effect of Regularizers was also observed in Figure 13 where Mish shows consistent improvement over Swish while using L1 penalty of 0.01 and L1-L2 penalty of 0.01; and matches the score of Swish while using L2 penalty of 0.01.

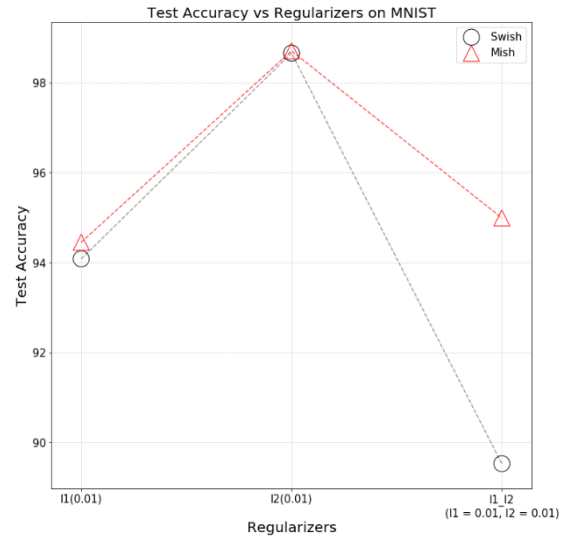

Figure 13. Test Accuracy v/s Regularizers for Mish and Swish.

Dropout [16] is a fundamental process used in Neural Network Training to avoid Overfitting. In Figure 14, Mish is shown to have a consistent improvement over Swish using different Dropout rates from 0.2 to 0.75 for a single dropout layer in a 4-layered network.

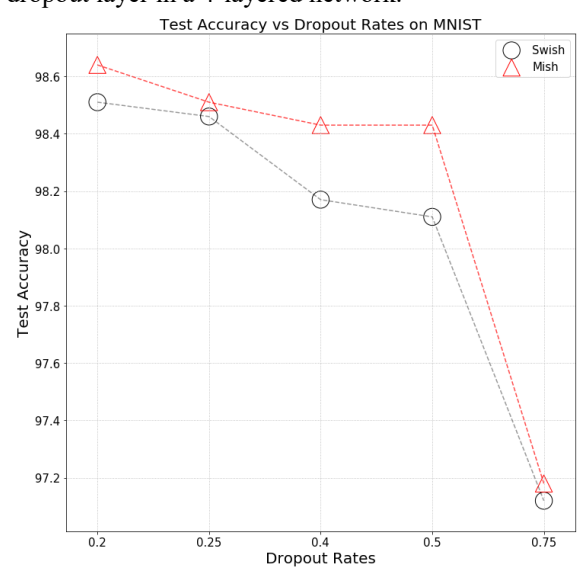

Figure 14. Test Accuracy v/s Dropout Rates for Mish and Swish.

Mish also shows improved results over Swish with increasing width of the network. In Figure 15, the number of neurons in the single dense layer of the 4-layered MNIST classifier was increased from 100 to 1024 and the resultant Test accuracy on MNIST dataset classification for Mish and Swish was observed and compared which showcases significant accuracy improvements for Mish as compared to the network initialized with Swish.

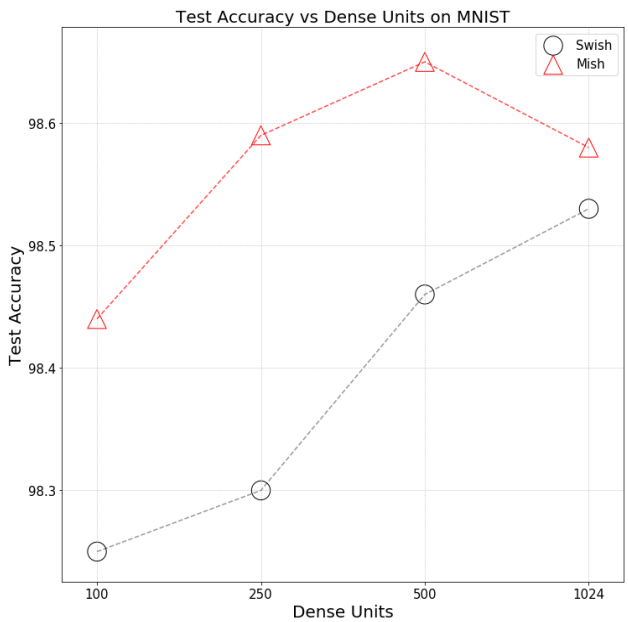

Figure 15. Test Accuracy v/s Width of Network for Mish and Swish.

3.2. CIFAR-10

To prove the consistency of Mish against Swish, the same experiments of hyper-parameter variation as discussed in section 3.1 for MNIST classification task was also observed in a 6-layered Convolutional Neural Network for classification of CIFAR-10 which are demonstrated in Figures 16-21. The results show a consistent performance increase of using Mish as compared to Swish. Certain results showcase significant accuracy increase for instance as shown in Figure 18 which compares the Test Accuracy for Mish and Swish while using different Initializers, it is observed while using Truncated Normal Initializer for the network with Mish results in ≈ 4% increase in Test Accuracy as compared to that of Swish. Additionally, it was observed that the network with Mish using He Initializer performed the best as compared to all other Initializers with a Test accuracy recorded to be at ≈ 75.6% while for the same, the network with Swish recorded a test accuracy of ≈ 71.7%.

In Figure 21, it is observed Mish significantly outperforms Swish while using NADAM [18,19] optimizer. The network with Swish seemed to suffer from gradient death where the network doesn't seem to learn at all throughout its training process of 10 epochs.

The similar trends in results from these experiments across both CIFAR-10 classification task and MNIST classification suggests Mish to be consistent and robust with different parameters commonly used in Neural Network training.

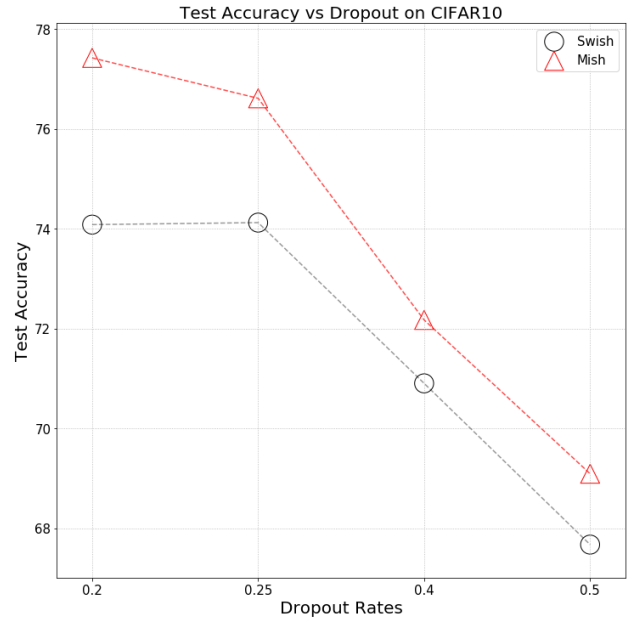

Figure 16. Test Accuracy v/s Dropout Rates for Mish and Swish.

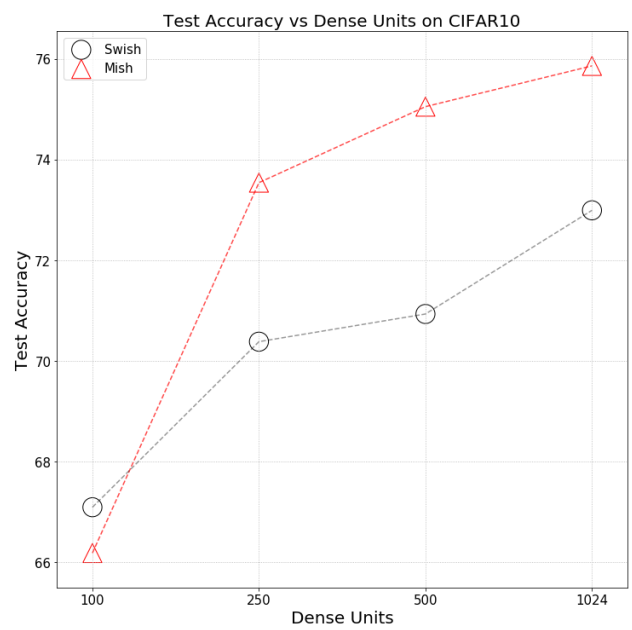

Figure 17. Test Accuracy v/s Width of the Network for Mish and Swish.

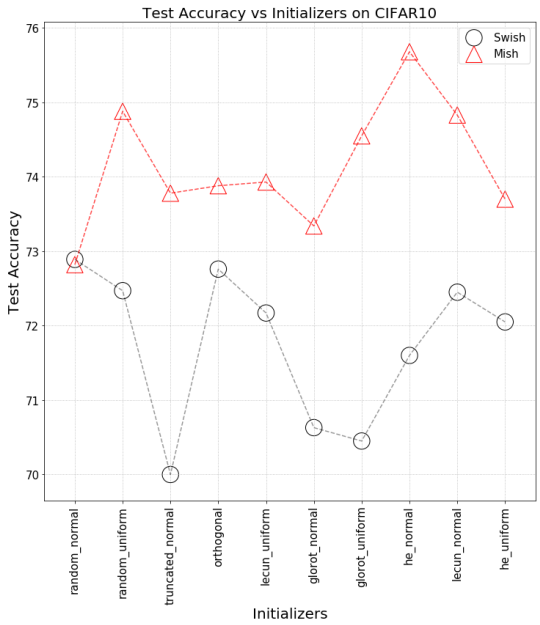

Figure 18. Test Accuracy v/s Initalizers for Mish and Swish.

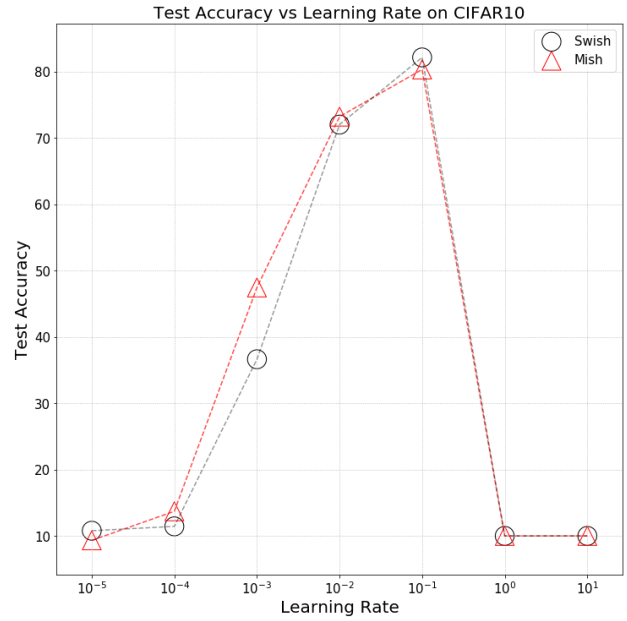

Figure 20. Test Accuracy v/s Learning Rates for Mish and Swish.

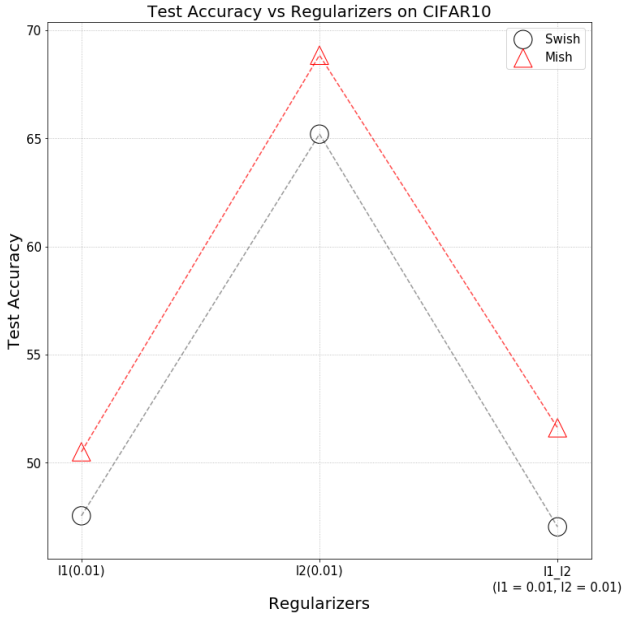

Figure 19. Test Accuracy v/s Regularizers for Mish and Swish.

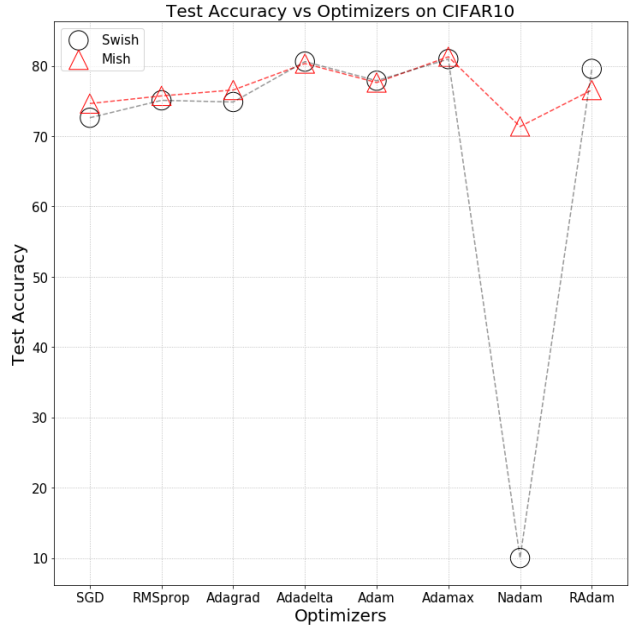

Figure 21. Test Accuracy v/s Optimizers for Mish and Swish.

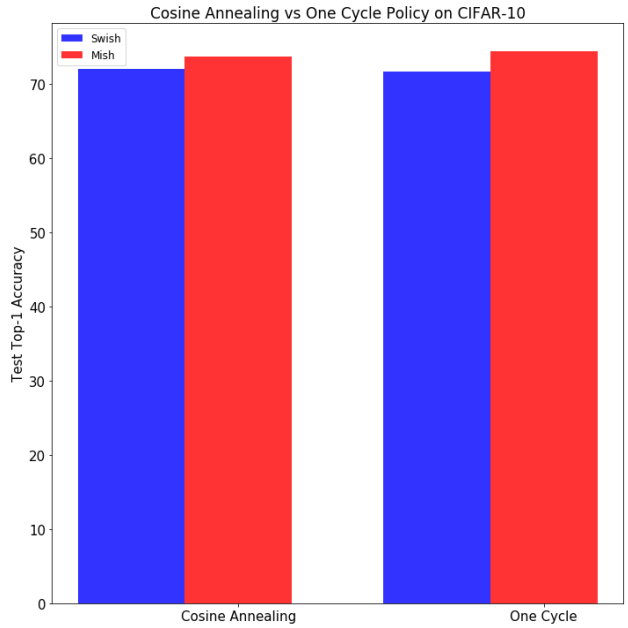

Figure 22. Cosine Annealing v/s One Cycle Policy on Test Accuracy for Mish and Swish.

Cosine Annealing [20] of Learning Rate is a process of Learning Rate decay during the training process of Neural Networks using a cosine function on an initial learning rate. In Figure 22, Mish was compared to Swish while using Cosine Annealing on a 6-layered network for CIFAR-10 classification task. For the experiment, max η was set at 0.01 (1e-2) and minimum η was set at 0.0001 (1e-4).

Figure 22 also demonstrates the comparison analysis between Mish and Swish while using One Cycle Policy [21,22]. One Cycle Policy is a process of finding the optimal learning rate during the training of a neural network where during the first half of the cycle, the learning rate is gradually increased with the corresponding decrease in momentum while in the second half of the cycle, the learning rate is gradually decreased with the simultaneous increase in momentum. In both these experiments, Mish outperformed Swish with higher Test Accuracy. For One Cycle Policy, Minimum Learning Rate was set at 7e-3, Max Learning Rate was set at 7e-2, Minimum Momentum was set at 0.85, Max Momentum was set at 0.95, Annealing Stage was set at 0.1 and Annealing Rate was set at 0.01.

Augmentation is considered to be an efficient hack in improving Neural Network performance especially in scenarios where the size of training data is considerably small. Mixup [23] is a simple image augmentation technique that mixes up both the training images and labels by linear interpolation using a weight parameter λ which is drawn from a Beta Distribution using a hyper-parameter α.

In both Figure 23 and Figure 24, Mish is compared with ReLU with Mixup of varying α values and the test loss and corresponding Test Accuracy. As observed, Mish consistently outperformed ReLU with a ≈ 9% increase in Test Accuracy while using Mixup at α = 0.2.

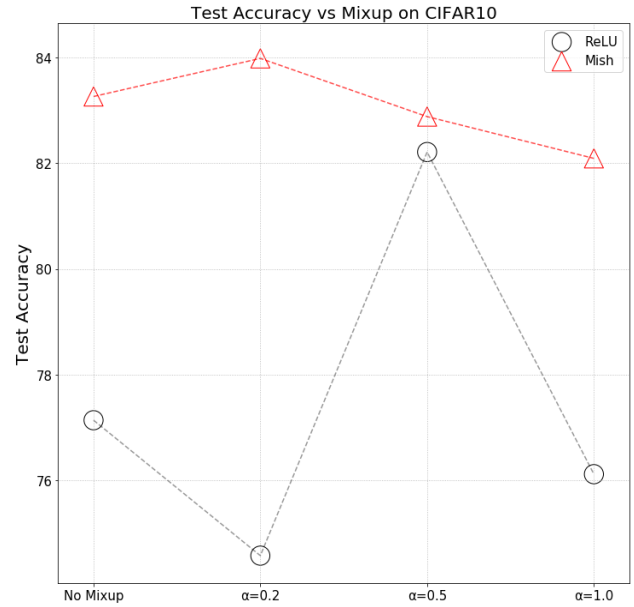

Figure 23. Test Accuracy v/s Mixup for Mish and ReLU.

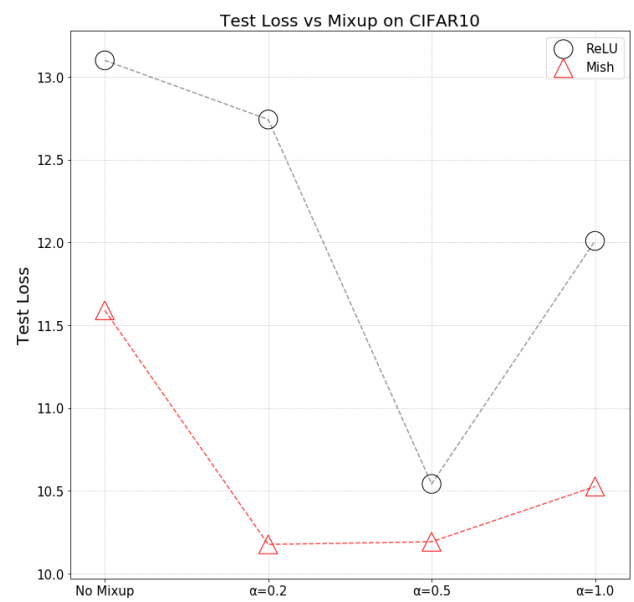

Figure 24. Test Loss v/s Mixup for Mish and Swish.

4. Statistical Analysis

To further evaluate the consistency of Mish as compared to other activation functions, the p-values and

confidence interval scores along with the mean test accuracy, mean test loss and standard deviation of test accuracy was noted for CIFAR-10 classification using a Squeeze Net for 23 runs with each run of 50 epochs using Adam [24,25] optimizer.

Table 2. Statistical Test Summary

Activation Function	Mean Accuracy	Mean Loss	Standard Deviation (Accuracy)	P-value
Mish	**87.48%**	**4.13%**	**0.3967**	-
Swish	87.32%	4.22%	0.414	0.197
GELU	87.37%	4.339%	0.472	0.4
ReLU	86.66%	4.398%	0.584	<1e-4
ELU	86.41%	4.211%	0.3371	<1e-4
Leaky ReLU	86.85%	4.112%	0.4569	<1e-4
SELU	83.91%	4.831%	0.5995	<1e-4
SoftPlus	83.004%	5.546%	1.4015	<1e-4
ReLU6	86.75%	4.355%	0.4501	<1e-4
SReLU	85.05%	4.541%	0.5826	<1e-4
ISRU	86.85%	4.669%	0.1106	<1e-4
LeCun's Tanh	82.72%	5.322%	0.58256	<1e-4
RReLU	86.87%	4.138%	0.4478	<1e-4
ELisH	87.38%	4.288%	0.47731	0.428

As Table 2 demonstrates, Mish has the highest mean test accuracy, lowest mean test loss, lowest standard deviation on the test accuracy and beats 10 out of the 13 activation functions at a high significance level ($P < 0.0001$).

The Confidence Intervals at a confidence level of 95% were also calculated for the different activation functions as shown in both Figure 25 and Table 3 using the results from the 23 runs used for P-value calculation shown in Table 2. Mish shows a better confidence interval profile with a higher mean and a smaller interval width as compared to most other commonly used activation functions.

Table 3. Confidence Interval Profile

Activation Function	95% CI
Mish	87.48 ± 0.1716
Swish	87.32347 ± 0.179027
GELU	87.37083 ± 0.2040073
ReLU	86.65869 ± 0.2524601
ELU	86.40565 ± 0.1458006
Leaky ReLU	86.84826 ± 0.1976138
SELU	83.91086 ± 0.2592722
SoftPlus	83.00434 ± 0.6060631
ReLU6	86.75391 ± 0.1946326
SReLU	85.04565 ± 0.2519697
ISRU	83.44652 ± 0.4323568
LeCun's Tanh	82.71956 ± 0.2519178
RReLU	86.86913 ± 0.1936264
ELisH	87.37652 ± 0.2064078

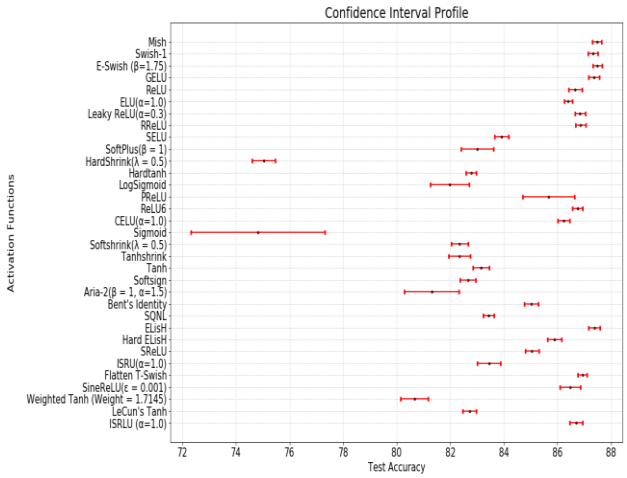

Figure 25. Confidence Interval Profiles of different activation functions.

5. Experiments

In total, at the time of this paper publication, Mish activation function has been validated on more than 70 benchmarks in problems ranging from Image Classification, Image Segmentation, Entity Embedding and Generative Models against a total of 35 activation functions on challenging datasets including CIFAR-10, CIFAR-100, CalTech-256 and ASL to name a few. Table 4 provides a summary of these results.

It is also to be noted that being more computationally expensive, Mish usually takes more time per epoch as compared to Swish and ReLU. However, a CUDA version of Mish is available which shows significant improvement in both forward pass and backward pass timings and is comparable to that of ReLU but is only limited for GPU inference.

Comparison is done based on the high priority metric, for image classification the Top-1 Accuracy while for Generative Networks and Image Segmentation the Loss Metric. Therefore, for the latter, "Mish > Baseline" is indicative of better loss and vice versa.

Table 4 is a concrete and conclusive proof of the robustness and efficiency of Mish activation function as compared to other baseline activation functions especially Swish and ReLU, which, as observed is outperformed 55 times across 75 benchmark experiments for ReLU and 53 times across the 75 benchmark experiments for Swish. These benchmarks were performed using a variety of models including DenseNet [26], Inception v3 [27], Xception Net [28] amongst many others. All results were based on the best of 3 runs. All the benchmarks for fair

comparison were made by just changing the activation function in the network architecture while keeping all other hyper-parameters to be constant.

Table 4. Benchmark Summary of Mish

Activation Function	Mish > Baseline	Mish < Baseline
ReLU	**55**	20
Swish	**53**	22
SELU	**21**	1
TanH	**19**	0
Sigmoid	**19**	0
Softsign	**19**	1
ELU	**19**	4
Tanhshrink	**18**	0
Hardshrink	**18**	0
Softshrink	**18**	0
PReLU	**18**	2
E-Swish	**18**	6
Hardtanh	**17**	1
GELU	**17**	2
CELU	**16**	2
LogSigmoid	**16**	3
Softplus	**15**	4
ReLU6	**14**	5
Leaky ReLU	**13**	7
Aria-2	**12**	1
RReLU	**11**	8
SQNL	**10**	0
Bent's Identity	**8**	2
Hard ELisH	**7**	0
Soft Clipping	**7**	0
LeCun's Tanh	**7**	0
SineReLU	**7**	1
Flatten T-Swish	**7**	1
ISRU	**6**	0
Weighted Tanh	**6**	0
ELisH	**6**	1
ISRLU	**4**	2
SReLU	**4**	2
Hard Sigmoid	**1**	0
Threshold ReLU	**1**	0

5.1. CIFAR

Table 5. CIFAR-10 Results (Test Top-1 Accuracy)

Model	Mish	Swish	ReLU
ResNet v2-20	**92.02%**	91.61%	91.71%
WRN 10-2	**86.83%**	86.56%	84.56%
SimpleNet	**91.70%**	91.44%	91.16%
Xception Net	**88.73%**	88.56%	88.38%
Capsule Net	**83.15%**	82.48%	82.19%
Inception ResNet v2	**85.21%**	84.96%	82.22%
DenseNet-121	**91.27%**	90.92%	91.09%
Mobile Net v2	**86.25%**	86.08%	86.05%
Shuffle Net v1	**87.31%**	86.95%	87.04%
Inception v3	**91.19%**	91.17%	90.84%
Efficient Net B0	**80.73%**	79.37%	79.31%

Table 6. CIFAR-100 Results (Test Top-1 Accuracy)

Model	Mish	Swish	ReLU
ResNet v2-110	**74.41%**	74.13%	73%
WRN 22-10	**72.32%**	71.89%	72.2%
WRN 40-4	69.52%	**69.59%**	69.35%
DenseNet - 121	**66.31%**	65.91%	65.50%
DenseNet - 169	65.38%	**65.69%**	64.99%
ResNext-50	**67.58%**	66.72%	67.52%
MobileNet v1	**50.09%**	49.95%	49.20%
SE Net-18	**64.38%**	63.89%	62.71%
Shuffle Net v2	**59.35%**	58.91%	58.56%
Squeeze Net	**63.07%**	62.11%	60.92%

Table 5 and 6 provide Top-1 Test accuracy stats of various models trained on Mish, Swish and ReLU. For the reduction of reading complexity, a small fraction of the experiment results out of all the benchmarks conducted is shown. As seen in both the tables, Mish consistently outperforms both ReLU and Swish with higher Top-1 testing accuracy while keeping every other parameter of the model to be constant.

Figure 26 shows the Training Graph of SENet-50 on CIFAR 10 using Mish, Swish, GELU, SELU, E-Swish and ReLU. As shown in the figure, Mish has a higher epoch time as compared to ReLU and Swish which can be commented as one of its drawbacks in comparison to other commonly used baseline activation functions.

Figure 26 also demonstrates the smoothness/ increased stability of the loss graph of Mish, an observation noticed in mostly all the tasks Mish has been benchmarked on.

All conducted and to-be-conducted benchmarks/ statistical tests are available on the GitHub repository (https://github.com/digantamisra98/Mish) of this paper.

6. Future Work

Mish is constantly being evaluated on various tasks and has successfully helped in achieving several top leaderboard scores. Mish paired with Ranger Optimizer [29] was used in securing 12 top leaderboard scores on the Fast AI ImageNette and ImageWoof benchmarks. Efficient Net B3 [30] model with Mish outperformed baseline Efficient Net B3 model's score on Stanford Cars Dataset classification task. Future work of Mish includes to benchmark it on ImageNet, NLP problems like Machine Translation and also to extend the theoretical understanding of Mish by comparing its rate of convergence with that of other activation functions.

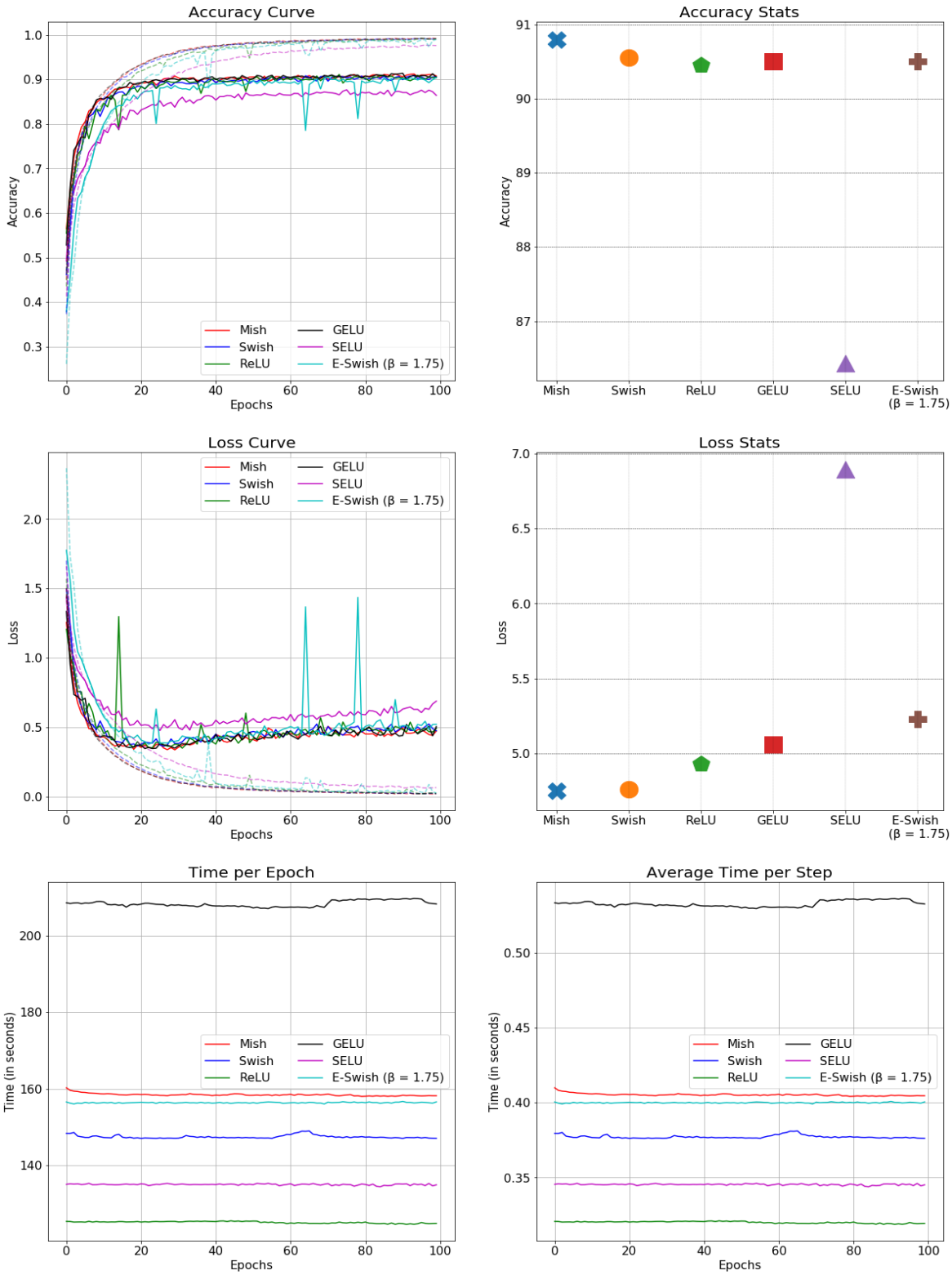

Figure 26. Training Graph of SENet-50 on CIFAR-10 using different activation functions along with Mish.

7. Conclusion

Mish is a novel neural activation function defined by the formula: $f(x) = x \cdot tanh(softplus(x))$. Being a non- monotonic, self-gated/regularized, smooth activation function; it demonstrates its capability and robustness in improving the results of a Neural Network task as compared to Swish and ReLU. The experiments concluded had models specifically synthesized with parameters favorable for ReLU, where in-place of ReLU, Swish and Mish were added correspondingly to obtain the results. This is a proof of its efficiency and suggests even improved performances with networks initialized with hyperparameters for Mish. Although, there is a certain trade-off of higher epoch time as observed in Mish, the improved accuracy and with the availability of higher compute it becomes a robust choice to replace Swish and ReLU in a model with Mish activation function.

References

[1] 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. Nature, 405(6789):947, 2000.

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

[3] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In International Conference on Machine Learning, 2010.

[4] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Swish: a self-gated activation function." arXiv preprint arXiv:1710.05941 7 (2017).

[5] Hayou, Soufiane, Arnaud Doucet, and Judith Rousseau. "On the selection of initialization and activation function for deep neural networks." arXiv preprint arXiv:1805.08266 (2018).

[6] Wuraola, Adedamola, and Nitish Patel. "SQNL: A New Computationally Efficient Activation Function." In 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-7. IEEE, 2018.

[7] Xu, Bing, Naiyan Wang, Tianqi Chen, and Mu Li. "Empirical evaluation of rectified activations in convolutional network." arXiv preprint arXiv:1505.00853 (2015).

[8] Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7132-7141. 2018.

[9] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520. 2018.

[10] Liu, Qian, and Steve Furber. "Noisy Softplus: a biology inspired activation function." In International Conference on Neural Information Processing, pp. 405-412. Springer, Cham, 2016.

[11] Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In USENIX Symposium on Operating Systems Design and Implementation, volume 16, pp. 265–283, 2016.

[12] Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch." (2017).

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256, 2010.

[14] 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, pp. 770–778, 2016a.

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

[16] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15, no. 1 (2014): 1929-1958.

[17] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." In Proceedings of COMPSTAT'2010, pp. 177-186. Physica-Verlag HD, 2010.

[18] Dozat, Timothy. "Incorporating nesterov momentum into adam." (2016).

[19] Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning." In International conference on machine learning, pp. 1139-1147. 2013.

[20] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." arXiv preprint arXiv:1608.03983 (2016).

[21] Smith, Leslie N. "A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay." arXiv preprint arXiv:1803.09820 (2018).

[22] Smith, Leslie N., and Nicholay Topin. "Super-convergence: Very fast training of neural networks using large learning rates." In Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006, p. 1100612. International Society for Optics and Photonics, 2019.

[23] Zhang, Hongyi, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. "mixup: Beyond empirical risk minimization." arXiv preprint arXiv:1710.09412 (2017).

[24] Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond." arXiv preprint arXiv:1904.09237 (2019).

[25] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[26] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In Proceedings of the IEEE conference on

computer vision and pattern recognition, pp. 4700-4708. 2017.
[27] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.
[28] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258. 2017.
[29] Zhang, Michael R., James Lucas, Geoffrey Hinton, and Jimmy Ba. "Lookahead Optimizer: k steps forward, 1 step back." arXiv preprint arXiv:1907.08610 (2019).
[30] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." arXiv preprint arXiv:1905.11946 (2019).