Mish: A Self Regularized Non-Monotonic Activation Function

Diganta Misra
mishradiganta91@gmail.com

Landskape
KIIT, Bhubaneswar, India

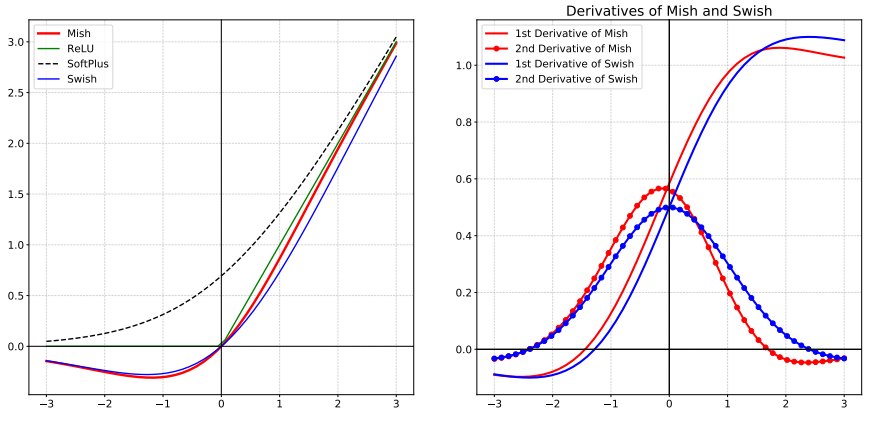
Abstract

We propose *Mish*, a novel self-regularized non-monotonic activation function which can be mathematically defined as: $f(x) = x\tanh(softplus(x))$. As activation functions play a crucial role in the performance and training dynamics in neural networks, we validated experimentally on several well-known benchmarks against the best combinations of architectures and activation functions. We also observe that data augmentation techniques have a favorable effect on benchmarks like ImageNet-1k and MS-COCO across multiple architectures. For example, Mish outperformed Leaky ReLU on YOLOv4 with a CSP-DarkNet-53 backbone on average precision ($AP_{50}{}^{val}$) by 2.1% in MS-COCO object detection and ReLU on ResNet-50 on ImageNet-1k in Top-1 accuracy by ≈1% while keeping all other network parameters and hyperparameters constant. Furthermore, we explore the mathematical formulation of Mish in relation with the Swish family of functions and propose an intuitive understanding on how the first derivative behavior may be acting as a regularizer helping the optimization of deep neural networks. Code is publicly available at https://github.com/digantamisra98/Mish.

1 Introduction

Activation functions are non-linear point-wise functions responsible for introducing non-linearity to the linear transformed input in a layer of a neural network. The choice of activation function is imperative for understanding the performance of a neural network. The process of applying an activation function in a layer of a neural network can be mathematically realized as $z = g(y) = g(\sum_i w_i x_i + b)$ where z is the output of the activation function $g(y)$. In early literature, Sigmoid and TanH activation functions were extensively used, which subsequently became ineffective in deep neural networks. A less probability inspired, unsaturated piece-wise linear activation known as Rectified Linear Unit (ReLU) [25, 34] became more relevant and showed better generalization and improved speed of convergence compared to Sigmoid and TanH.

Although ReLU demonstrates better performance and stability compared to TanH and Sigmoid, it is not without weaknesses. One of which is popularly known as Dying ReLU, which is experienced through a gradient information loss caused by collapsing the negative inputs to zero. Over the years, many activation functions have been proposed which improve performance and address the shortcomings of ReLU, which include Leaky ReLU [32], ELU [6], and SELU [23]. Swish [37], which can be defined as $f(x) = xsigmoid(\beta x)$, proved to be a more robust activation function showcasing strong improvements in results as compared to



(a) (b)

Figure 1: (a) Graph of Mish, ReLU, SoftPlus, and Swish activation functions. As illustrated, Mish and Swish are closely related with both having a distinctive negative concavity unlike ReLU, which accounts for preservation of small negative weights. (b) The 1^{st} and 2^{nd} derivatives of Mish and Swish activation functions.

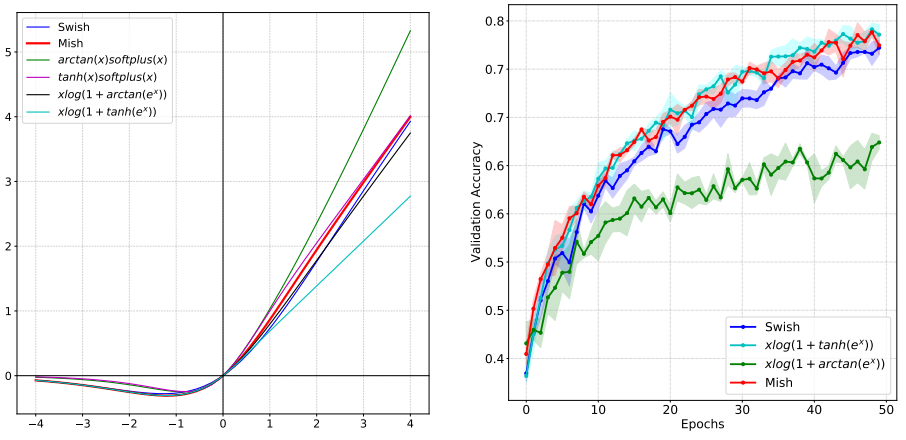
that of ReLU. The smooth, continuous profile of Swish proved essential in better information propagation as compared to ReLU in deep neural network architectures.

In this work, we propose **Mish**, a novel self regularized non-monotonic activation function inspired by the self gating property of Swish. Mish is mathematically defined as: $f(x) = x\tanh(softplus(x))$. We evaluate and find that Mish tends to match or improve the performance of neural network architectures as compared to that of Swish, ReLU, and Leaky ReLU across different tasks in Computer Vision.

2 Motivation

Across theoretical research into activation functions, those sharing properties similar to Swish, which includes non-monotonicity, ability to preserve small negative weights, and a smooth profile, have been a recurring discussion. For instance, Gaussian Error Linear Units (GELU) [16] is a popular activation function which has similar properties to that of Swish and is actively used in the GPT-2 architecture [36] for synthetic text generation. Swish was discovered by a Neural Architecture Search (NAS) [52] over the space of the non-linear functions by a controlled search agent. An RNN-controller was used as the agent which generated a new candidate function at each step, for a total of 10K steps, which were then evaluated on CIFAR-10 classification task using a ResNet-20 defined with that candidate function as its activation function. The design of Mish, while influenced by the work performed by Swish, was found by systematic analysis and experimentation over the characteristics that made Swish so effective. When studying similarly behaved functions like Swish, as illustrated in Fig. 2 (a), which include $\arctan(x)softplus(x)$, $\tanh(x)softplus(x)$, $x\log(1+\arctan(e^x))$ and $x\log(1+\tanh(e^x))$, where $softplus(x) = \ln(1+e^x)$, from our ablation study we determined Mish consistently outperforms the aforementioned functions along with Swish and ReLU.

We used a standard six-layered deep convolution neural network architecture to validate each of the experimental activation functions earlier defined on the CIFAR-10 image classi-



(a) (b)

Figure 2: (a) Graph of Mish, Swish, and similar validated experimental functions. (b) Training curve of a six-layered CNN on CIFAR-10 on different validated activation functions.

fication task. The networks were trained for three runs, each for 50 epochs with RMSProp as the optimizer. As shown in Fig. 2 (b), we found that Mish performed better than the other validated functions. Although it can be observed that $x\log(1+\tanh(e^x))$ performed at par to Mish, we noted that its training is often unstable and, in many cases, leads to divergence in deeper architectures. We observed similar unstable training issues for $\arctan(x)softplus(x)$ and $\tanh(x)softplus(x)$. While all of the validated functions have a similar shape, Mish proves to be consistently better in terms of performance and stability.

While not evident at first sight, Mish is closely related to Swish, as it can be observed in the first derivative:

$$f'(x) = sech^2(softplus(x))xsigmoid(x) + \frac{f(x)}{x} \tag{1}$$

$$= \Delta(x)swish(x) + \frac{f(x)}{x} \tag{2}$$

where $softplus(x) = \ln(1+e^x)$ and $sigmoid(x) = 1/(1+e^{-x})$.

From experimental observations, we speculate that the $\Delta(x)$ parameter acts like a preconditioner, making the gradient smoother. Preconditioning has been extensively discussed and used in general optimization problems where the preconditioner, in case of gradient descent [3, 25] is the inverse of a symmetric positive definite matrix (H_k^{-1}) which is applied to modify the geometry of the objective function to increase the rate of convergence [1]. Intuitively, preconditioning makes the objective function much smoother and thus making it easier to optimize. The $\Delta(x)$ parameter mimics the behavior of a preconditioner. It provides a strong regularization effect and helps make gradients smoother, which corresponds to easier to optimize function contour, which is possibly why Mish outperforms Swish in increasingly deep and complex neural net architectures.

Mish, additionally, similar to Swish, is non-monotonic, smooth, and preserves a small amount of negative weights. These properties account for the consistent performance and improvement when using Mish in-place of Swish in deep neural networks.

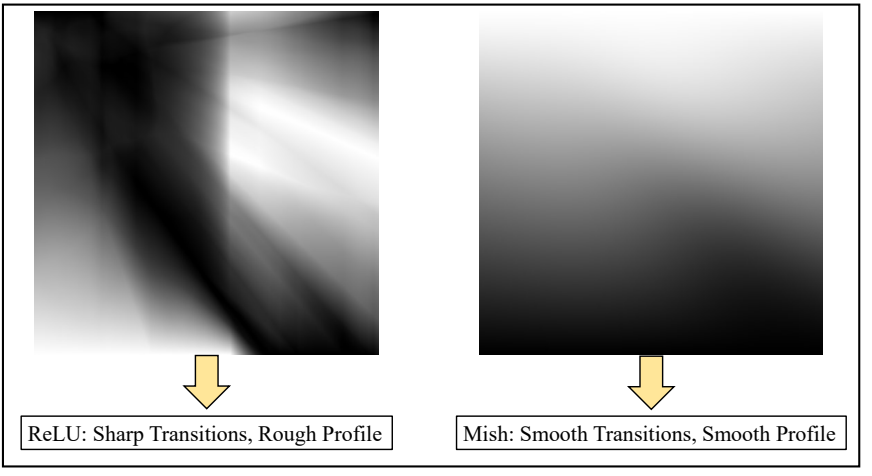


Figure 3: Comparison between the output landscapes of ReLU and Mish activation function

3 Mish

Mish, as visualized in Fig. 1 (a), is a smooth, continuous, self regularized, non-monotonic activation function mathematically defined as:

$$f(x) = x\tanh(softplus(x)) = x\tanh(\ln(1+e^x)) \tag{3}$$

Similar to Swish, Mish is bounded below and unbounded above with a range of [$\approx$ -0.31, ∞). The 1^{st} derivative of Mish, as shown in Fig. 1 (b), can be defined as:

$$f'(x) = \frac{e^x \omega}{\delta^2} \tag{4}$$

where, $\omega = 4(x+1) + 4e^{2x} + e^{3x} + e^x(4x+6)$ and $\delta = 2e^x + e^{2x} + 2$. Inspired by Swish, Mish uses the Self-Gating property where the non-modulated input is multiplied with the output of a non-linear function of the input. Due to the preservation of a small amount of negative information, Mish eliminated by design the preconditions necessary for the Dying ReLU phenomenon. This property helps in better expressivity and information flow. Being unbounded above, Mish avoids saturation, which generally causes training to slow down due to near-zero gradients [11] drastically. Being bounded below is also advantageous since it results in strong regularization effects. Unlike ReLU, Mish is continuously differentiable, a property that is preferable because it avoids singularities and, therefore, undesired side effects when performing gradient-based optimization.

Having a smooth profile also plays a role in better gradient flow, as shown in Fig. 3, where the output landscapes of a five-layered randomly initialized neural network with ReLU and Mish are visualized. The landscapes were generated by passing in the co-ordinates to a five-layered randomly initialized neural network which outputs the corresponding scalar magnitude. The output landscape of ReLU has a lot of sharp transitions as compared to the smooth profile of the output landscape of Mish. Smoother output landscapes suggest smooth loss landscapes [28], which help in easier optimization and better generalization, as demonstrated in Fig. 4.

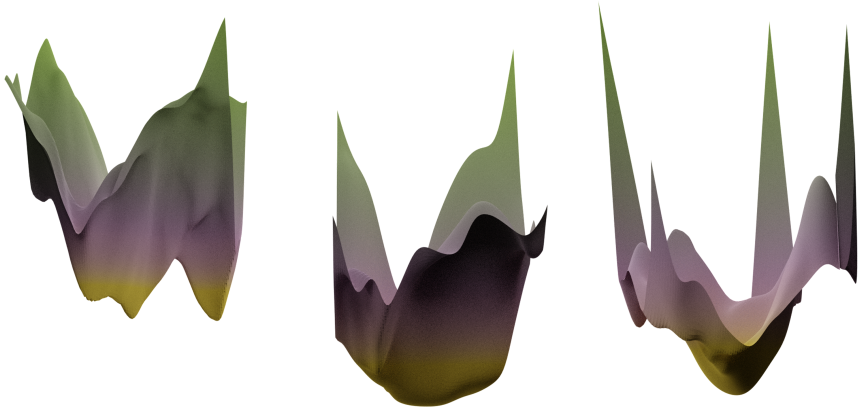


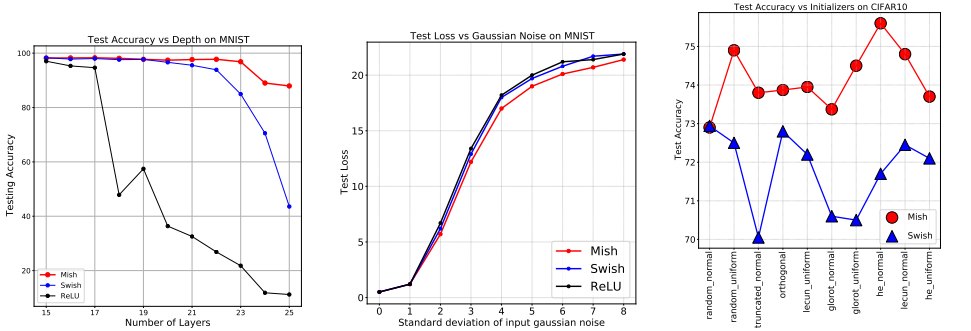
Figure 4: Comparison between the loss landscapes of (from left to right): (a) ReLU, (b) Mish and (c) Swish activation function for a ResNet-20 trained for 200 epochs on CIFAR-10.

We observed the loss landscapes [28] of a ResNet-20 [15] equipped with ReLU, Mish, and Swish activation functions with each trained for 200 epochs for the image classification task on the CIFAR-10 dataset. We used a multi-step learning rate policy with the SGD optimizer for training the networks. As shown in Fig. 4, the loss landscape for the ResNet-20 equipped with Mish is much smoother and conditioned as compared to that of ReLU and Swish activation function. Mish has a wider minima which improves generalization compared to that of ReLU and Swish, with the former having multiple local minimas. Additionally, Mish obtained the lowest loss as compared to the networks equipped with ReLU and Swish, and thus, validated the preconditioning effect of Mish on the loss surface.

3.1 Ablation Study on CIFAR-10 and MNIST

Hyperparameters, including the depth of the network, type of weight initialization, batch size, learning rate, and optimizer used in the training process, have significant unique effects. We manipulate different hyper-parameters to observe their effects on the performance of ReLU, Swish, and Mish activation functions. Firstly, we observe the effect of increasing the number of layers of a neural network with ReLU, Swish, and Mish on the test accuracy. For the task, we used the MNIST dataset [26] and trained fully connected networks of linearly increasing depth. Each layer was initialized with 500 neurons, while Residual Units [15] were not used since they allow the training of arbitrary deep networks. We used Batch Normalization [20] layers to decrease the dependence on initialization along with Dropout [41] of 25%. The network was optimized using SGD [3] with a batch size of 128. For a fair comparison, the same learning rate was maintained for the three networks with ReLU, Swish, and Mish. As shown in Fig. 5 (a), post fifteen layers, there was a sharp decrease in accuracy for both Swish and ReLU, while Mish maintained a significantly higher accuracy in large models where optimization becomes difficult. This property was later validated in ImageNet-1k [7] experiments in Section 4.3, where Mish performed superior to Swish in increasingly large networks.

We also evaluated the robustness of Mish in noisy input conditions where the input MNIST data was corrupted with additive zero-centered Gaussian Noise with linearly in-



(a) (b) (c)

Figure 5: (a) Comparison between Mish, Swish, and ReLU activation functions in terms of test accuracy with increasing depth of the neural network on the MNIST dataset. (b) Comparison between Mish, Swish, and ReLU activation functions in terms of test loss with increasing input gaussian noise on the MNIST dataset. (c) Comparison between Mish and Swish activation functions in terms of test accuracy with different weight initialization strategies on the CIFAR-10 dataset.

creasing standard deviation. We used a five-layered convolution neural network architecture optimized using SGD for this task. Fig. 5 (b) demonstrates the consistently better loss with varying intensity of Input Gaussian Noise with Mish as compared to ReLU and Swish.

Initializers [11] play a crucial role in the performance of a neural network. We observed the performance of Mish and Swish using different weight initializers, including Glorot initializer [11], LeCun normal initializer [27], and He uniform variance scaling initializer [13], in a six-layered convolution neural network. Fig. 5 (c) demonstrates the consistent positive difference in the performance of Mish compared to Swish while using different initializers.

4 Benchmarks

We evaluated Mish against more than ten standard activation functions on different models and datasets. Our results show, especially in computer vision tasks like image classification and object detection, Mish consistently matched or exceeded the best performing network. We also recorded multiple runs to observe the statistical significance of our results. Along with the vanilla settings, we also validated the performance of Mish when coupled with various state of the art data augmentation techniques like CutMix and label smoothing.

4.1 Statistical Analysis

To evaluate the statistical significance and consistency of the performance obtained by Mish activation function compared to baseline activation functions, we calculate and compare the mean test accuracy, mean test loss, and standard deviation of test accuracy for CIFAR-10 [24] classification task using a Squeeze Net [18]. We experimented for 23 runs, each for 50 epochs using the Adam optimizer [22] and changing the activation functions while keeping every other network parameter constant. Table. 1 shows Mish outperforms other activation functions with the highest mean accuracy (μ_{acc}), second-lowest mean loss (μ_{loss}), and third-lowest standard deviation of accuracy (σ_{acc}).

Activation	μ_{acc}	μ_{loss}	σ_{acc}
Mish	**87.48**%	4.13%	0.3967
Swish [37]	87.32%	4.22%	0.414
GELU [16]	87.37%	4.339%	0.472
ReLU [25, 34]	86.66%	4.398%	0.584
ELU [6]	86.41%	4.211%	0.3371
Leaky ReLU [32]	86.85%	**4.112**%	0.4569
SELU [23]	83.91%	4.831%	0.5995
SoftPlus	83%	5.546%	1.4015
SReLU [21]	85.05%	4.541%	0.5826
ISRU [4]	86.85%	4.669%	**0.1106**
TanH	82.72%	5.322%	0.5826
RReLU [48]	86.87%	4.138%	0.4478

Table 1: Statistical results of different activation functions on image classification of CIFAR-10 dataset using a Squeeze Net for 23 runs.

4.2 CIFAR-10

We compare the performance of different baseline activation functions on the image classification task of CIFAR-10 dataset [24] using different standard neural network architectures by just swapping the activation functions and keeping every other network parameter and training parameter constant. We evaluate the performance of Mish as compared to ReLU and Swish on various standard network architectures, including Residual Networks [15], Wide Residual Networks [50], Shuffle Net [51], Mobile Nets [17], Inception Network [42], and Efficient Networks [43]. Table. 2 shows that Mish activation function consistently outperforms ReLU and Swish activation functions across all the standard architectures used in the experiment, with often providing 1% to 3% performance improvement over the baseline ReLU enabled network architectures.

Architecture	Mish	Swish	ReLU
ResNet-20 [15]	**92.02**%	91.61%	91.71%
WRN-10-2 [50]	**86.83**%	86.56%	84.56%
SimpleNet [12]	**91.70**%	91.44%	91.16%
Xception Net [5]	**88.73**%	88.56%	88.38%
Capsule Net [40]	**83.15**%	82.48%	82.19%
Inception ResNet v2 [42]	**85.21**%	84.96%	82.22%
DenseNet-121 [19]	**91.27**%	90.92%	91.09%
MobileNet-v2 [17]	**86.25**%	86.08%	86.05%
ShuffleNet-v1 [51]	**87.31**%	86.95%	87.04%
Inception v3 [42]	**91.19**%	91.17%	90.84%
Efficient Net B0 [43]	**80.73**%	79.37%	79.31%

Table 2: Comparison between Mish, Swish, and ReLU activation functions based on test accuracy on image classification of CIFAR-10 across various network architectures.

4.3 ImageNet-1k

Additionally, we compare Mish with Leaky ReLU [32] and Swish for ImageNet 2012 dataset classification task. ImageNet [7] is considered to be one of the most challenging and significant classification tasks in the domain of computer vision. ImageNet comprises of 1.28 million training images distributed across 1,000 classes. We use the validation set comprising of 50,000 images to evaluate the performance of the trained networks. We trained the networks using the DarkNet framework [38] on an AWS EC2 p3.16xlarge instance comprising of 8 Tesla V100 GPUs for a total number of 8 million training steps with batch size, mini-batch size, initial learning rate, momentum, and weight decay set at 128, 32, 0.01, 0.9, and 5e-4 respectively.

Model	Data Augmentation	LReLU/ ReLU †		Swish		Mish	
		Top-1	Top- 5	Top-1	Top- 5	Top-1	Top- 5
ResNet-18 [15]	No	69.8%†	89.1%†	**71.2%**	**90.1%**	**71.2%**	89.9%
ResNet-50 [15]	No	75.2%†	92.6%†	75.9%	92.8%	**76.1%**	**92.8%**
SpineNet-49 [8]	Yes	77.0%†	93.3%†	78.1%	94%	**78.3%**	**94.6%**
PeleeNet [46]	No	70.7%	90.0%	**71.5%**	**90.7%**	71.4%	90.4%
CSP-ResNet-50 [45]	Yes	77.1%	94.1%	-	-	**78.1%**	**94.2%**
CSP-DarkNet-53 [7]	Yes	77.8%	94.4%	-	-	**78.7%**	**94.8%**
CSP-ResNext-50 [45]	No	77.9%	94.0%	64.5%	86%	**78.9%**	**94.5%**
CSP-ResNext-50 [45]	Yes	78.5%	94.8%	-	-	**79.8%**	**95.2%**

Table 3: Comparison between Mish, Swish, ReLU and Leaky ReLU activation functions on image classification of ImageNet-1k dataset across various standard architectures. Data Augmentation indicates the use of CutMix, Mosaic, and Label Smoothing. † indicate scores for ReLU.

In Table. 3, we compare the Top-1 and Top-5 accuracy of Mish against ReLU, Leaky ReLU, and Swish on PeleeNet [46] , Cross Stage Partial ResNet-50 [45], and ResNet-18/50 [15]. Mish consistently outperforms the default Leaky ReLU/ ReLU on all the four network architectures with a 1% increase in Top-1 Accuracy over Leaky ReLU in CSP-ResNet-50 architecture. Although Swish provides marginally stronger result in PeleeNet as compared to Mish, we investigate further of the inconsistency of the performance of Swish in a larger model where we compare Swish, Mish and ReLU in a CSP-ResNext-50 model [45, 47] where Swish decreases the Top-1 accuracy by 13.4% as compared to Leaky ReLU while Mish improves the accuracy by 1%. This shows that Swish cannot be used in every architecture and has drawbacks in especially large complex models like ResNext based models. We also combine different data augmentation techniques like CutMix [49] and Label Smoothing (LS) [33] to improve the baseline scores of CSP-ResNet-50, CSP-DarkNet-53 [7] and CSP-ResNext-50 models. The results suggest that Mish is more consistent and generally guarantees performance increase in almost any neural network for ImageNet classification.

4.4 MS-COCO Object Detection

Object detection [10] is a fundamental branch of computer vision that can be categorized as one of the tasks under visual scene understanding. In this section, we present our experimental results on the challenging Common Objects in Context (MS-COCO) dataset [30]. We report the mean average precision (mAP-50/ mAP@0.5) on the COCO test-dev split, as demonstrated in Table. 4. We report our results for two models, namely, CSP-DarkNet-53

[2] and CSP-DarkNet-53+PANet+SPP [2, 14], where we retrained the backbone network from scratch by replacing the activation function from ReLU to Mish.

We also validate our results by using various data augmentation strategies, including CutMix [49] , Mosaic [2], self adversarial training (SAT) [2], Dropblock regularization [9] and Label Smoothing [33] along with Mish. As per the results demonstrated in Table. 4, simply replacing ReLU with Mish in the backbone improved the mAP@0.5 for CSP-DarkNet-53 and CSP-DarkNet-53+PANet+SPP by 0.4%. For CSP-DarkNet-53, we achieve state of the art mAP@0.5 of 65.7% at a real-time speed of 65 FPS on Tesla V100. Additionally, CSP-DarkNet-53 was used as the backbone with a Yolov3 detector [39] as its object detection head. We use multi-input weighted residual connections (MiWRC) [24] in the backbone and train the model with a cosine annealing scheduler [31]. We also eliminate grid sensitivity and use multiple anchors for single ground truth for the detector. Experiments were done on a single GPU to enable multi-scale training with default parameters, including epochs, initial learning rate, weight decay, and momentum set at 500500, 0.01, 5e-4, and 0.9, respectively.

Model	Size	Data Augmentation	ReLU	Mish
CSP-DarkNet-53 [2]	(512 x 512)	No	64.5%	**64.9%**
CSP-DarkNet-53 [2]	(608 x 608)	No	-	**65.7%**
CSP-DarkNet53+PANet+SPP [2, 14]	(512 x 512)	Yes	64.5%	**64.9%**

Table 4: Comparison between ReLU and Mish activation functions on object detection on MS-COCO dataset.

We provide further comparative results using the YOLOv4 [2] detector, as demonstrated in Table. 5. Using Mish, we observed a consistent 0.9% to 2.1% improvement in the $AP_{50}{}^{val}$ on test size of 736. We evaluated three variants of YOLOv4, which are: $YOLOv4_{pacsp}$, $YOLOv4_{pacsp\text{-}s}$, and $YOLOv4_{pacsp\text{-}x}$. All three variants use a CSP-DarkNet-53 [45] and CSP-PANet in the backbone coupled with a CSP-SPP [14] (Spatial Pyramid Pool) module where the latter two variants denote the tiny and extra-large variant of $YOLOv4_{pacsp}$.

Detector	Activation	AP^{val}	$AP_{50}{}^{val}$	$AP_{75}{}^{val}$	$AP_S{}^{val}$	$AP_M{}^{val}$	$AP_L{}^{val}$
$YOLOv4_{pacsp\text{-}s}$	Leaky ReLU	36.0%	54.2%	39.4%	18.7%	41.2%	48.0%
	Mish	**37.4%**	**56.3%**	**40.0%**	**20.9%**	**43.0%**	**49.3%**
$YOLOv4_{pacsp}$	Leaky ReLU	46.4%	64.8%	**51.0%**	28.5%	51.9%	**59.5%**
	Mish	**46.5%**	**65.7%**	50.2%	**30.0%**	**52.0%**	59.4%
$YOLOv4_{pacsp\text{-}x}$	Leaky ReLU	47.6%	66.1%	52.2%	29.9%	53.3%	61.5%
	Mish	**48.5%**	**67.4%**	**52.7%**	**30.9%**	**54.0%**	**62.0%**

Table 5: Comparison between Leaky ReLU and Mish activation functions on object detection on MS-COCO 2017 dataset with a test image size of 736 x 736.

4.5 Stability, Accuracy, and Efficiency Trade-off

Mish is a novel combination of three activation functions, which are TanH, SoftPlus, and the identity function. In practical implementation, a threshold of 20 is enforced on Softplus, which makes the training more stable and prevents gradient overflow. Due to the increased complexity, there is a trade-off between the increase in accuracy while using Mish and the

increase in computational cost. We address this concern by optimizing Mish using a CUDA based implementation, which we call Mish-CUDA which is based on PyTorch [35].

Activation	Data Type	Forward Pass	Backward Pass
ReLU	fp16	$223.7\mu s \pm 1.026\mu s$	$312.1\mu s \pm 2.308\mu s$
SoftPlus	fp16	$342.2\mu s \pm 38.08\mu s$	$488.5\mu s \pm 53.75\mu s$
Mish	fp16	$658.8\mu s \pm 1.467\mu s$	$1.135ms \pm 4.785\mu s$
Mish-CUDA	fp16	$267.3\mu s \pm 1.852\mu s$	$345.6\mu s \pm 1.875\mu s$
ReLU	fp32	$234.2\mu s \pm 621.8ns$	$419.3\mu s \pm 1.238\mu s$
SoftPlus	fp32	$255.1\mu s \pm 753.6ns$	$420.2\mu s \pm 631.4ns$
Mish	fp32	$797.4\mu s \pm 1.094\mu s$	$1.689ms \pm 1.222\mu s$
Mish-CUDA	fp32	$282.9\mu s \pm 876.1ns$	$496.3\mu s \pm 1.781\mu s$

Table 6: Comparison between the runtime for the forward and backward passes for ReLU, SoftPlus, Mish and Mish-CUDA activation functions for floating point-16 and floating point-32 data.

In Table. 6, we show the speed profile comparison between the forward pass (FWD) and backward pass (BWD) on floating-point 16 (FP16) and floating-point 32 (FP32) data for ReLU, SoftPlus, Mish, and Mish-CUDA. All runs were performed on an NVIDIA GeForce RTX-2070 GPU using standard benchmarking practices over 100 runs, including warm-up and removing outliers.

Table. 6 shows the significant reduction in computational overhead of Mish by using the optimized version Mish-CUDA which shows no stability issues, mirrors the learning performance of the original baseline Mish implementation and is even faster than native PyTorch Softplus implementation in single precision, making it more feasible to use Mish in deep neural networks. Mish can be further optimized using the exponential equivalent of the TanH term to accelerate the backward pass, which involves the derivative computation.

5 Conclusion

In this work, we propose a novel activation function, which we call Mish. Even though Mish shares many properties with Swish and GELU like unbounded positive domain, bounded negative domain, non-monotonic shape, and smooth derivative, Mish still provides under most experimental conditions, better empirical results than Swish, ReLU, and Leaky ReLU. We expect that a hyperparameter search with Mish as a target may improve upon our results. We also observed that the state of the art data augmentation techniques like CutMix and other proven ones like Label Smoothing behave consistently with the expectations.

Future work includes optimizing Mish-CUDA to reduce the computational overhead further, evaluating the performance of the Mish activation function in other state of the art models on various tasks in the domain of computer vision, and obtaining a normalizing constant as a parameter for Mish which can reduce the dependency on using Batch Normalization layers. We believe it is of theoretical importance to investigate the contribution of the $\Delta(x)$ parameter at the first derivative and understand the underlying mechanism on how it may be acting as a regularizer. A clear understanding of the behavior and conditions governing this regularizing term could motivate a more principled approach to constructing better performing activation functions.

6 Acknowledgements

The author would like to dedicate this work to the memory of his late grandfather, Prof. Dr. Fakir Mohan Misra. The author would also like to offer sincere gratitude to everyone who supported during the timeline of this project including Sparsha Mishra, Alexandra Deis from X – The Moonshot Factory, Ajay Uppili Arasanipalai from University of Illinois - Urbana Champaign (UIUC), Himanshu Arora from Montreal Institute for Learning Algorithms (MILA), Javier Ideami, Federico Andres Lois from Epsilon, Alexey Bochkovskiy, Chien-Yao Wang, Thomas Brandon, Soumik Rakshit from DeepWrex, Less Wright, Manjunath Bhat from Indian Institute of Technology - Kharagpur (IIT-KGP), Miklos Toth and many more including the Fast.ai team, Weights and Biases community and everyone at Landskape.

References

[1] Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[4] Brad Carlile, Guy Delamarter, Paul Kinney, Akiko Marti, and Brian Whitney. Improving deep learning by inverse square root linear units (isrlus). *arXiv preprint arXiv:1710.09967*, 2017.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. *arXiv preprint arXiv:1912.05027*, 2019.

[9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018.

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[11] 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*, pages 249–256, 2010.

[12] Seyyed Hossein HasanPour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.

[13] 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.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[15] 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.

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

[19] Gao Huang, 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*, pages 4700–4708, 2017.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Xiaojie Jin, Chunyan Xu, Jiashi Feng, Yunchao Wei, Junjun Xiong, and Shuicheng Yan. Deep learning with s-shaped rectified linear activation units. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] 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.

[26] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[27] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[28] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[29] Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[31] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[32] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[33] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019.

[34] 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.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1 (8):9, 2019.

[37] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[38] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[39] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[40] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[41] Nitish Srivastava, 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(1):1929–1958, 2014.

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[43] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[44] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. *arXiv preprint arXiv:1911.09070*, 2019.

[45] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. Cspnet: A new backbone that can enhance learning capability of cnn. *arXiv preprint arXiv:1911.11929*, 2019.

[46] Robert J Wang, Xiang Li, and Charles X Ling. Pelee: A real-time object detection system on mobile devices. In *Advances in Neural Information Processing Systems*, pages 1963–1972, 2018.

[47] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

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

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

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

[51] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[52] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.