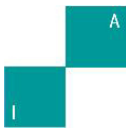
Inteligencia Artificial, 25(70) (2022), 95-109
doi: 10.4114/intartif.vol25iss70pp95-109



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

A Large-Scale Study of Activation Functions in Modern Deep Neural Network Architectures for Efficient Convergence

Andrinandrasana David Rasamoelina[1,A], Ivan Čik[1], Peter Sinčak[1,2], Marián Mach[1], Lukáš Hruška[1],

[1] Department of Cybernetics and Artificial intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic.
[A] andrijdavid@tuke.sk

[2] University of Miskolc, Faculty of Mechanical Engineering and Informatics. Institute of Informatics, Department of General Informatics, 3515 University City of Miskolc, Hungary.

Abstract Activation functions play an important role in the convergence of learning algorithms based on neural networks. They provide neural networks with nonlinear ability and the possibility to fit in any complex data. However, no deep study exists in the literature on the comportment of activation functions in modern architecture. Therefore, in this research, we compare the 18 most used activation functions on multiple datasets (CIFAR-10, CIFAR-100, CALTECH-256) using 4 different models (EfficientNet, ResNet, a variation of ResNet using the bag of tricks, and MobileNet V3). Furthermore, we explore the shape of the loss landscape of those different architectures with various activation functions. Lastly, based on the result of our experimentation, we introduce a new locally quadratic activation function namely Hytana alongside one variation Parametric Hytana which outperforms common activation functions and address the dying ReLU problem.

Keywords: Activation Function, Computer Vision, Deep Learning.

1 Introduction

Deep learning is used in every application field nowadays. Our current processing power enables us to process data at a large scale, and apply deep learning to almost every problem and data type *e.g.*, video, audio, text, signal, *etc*. We can even apply deep learning in multiple fields such as music [4, 30], fashion design [38, 46] or game design [13,53], video analysis [22,28], *etc*. But at the heart of all those applications reside deep learning architecture. Therefore studying and improving those architectures is important for the progress of artificial intelligence. In this study, we focus on the part of deep learning architecture that gives them the ability to model and fit complex data: The activation function. Activation functions play an important role in the convergence of learning algorithms based on neural networks. The nonlinear activation function is a central part that makes neural networks differ from linear models, *i.e*. a neural network becomes a linear function if the linear activation function is used. Therefore, the choice of a proper activation function substantially influences the performance and computational efficiency of neural networks. Numerous activation functions have been suggested to improve neural network performance *e.g.*: [16, 32, 36].

However, most of the studies focus on a particular type of activation function such as ReLU or a small class of activation functions. One of the most common and most used nowadays is ReLU. It is fast, provides good performance, and performs better than the sigmoidal activation function. However, this activation function also introduces the dying ReLU problem [31]. This problem occurs when neurons become inactive and only output

ISSN: 1137-3601 (print), 1988-3064 (on-line)

0 for any given input. In this study, we compared the 18 most used activation functions. Based on the result of our experiment, we introduce two novel activation functions in this work: Hytana (hyperbolic tangent activation function) and PHytana (parametric hyperbolic tangent activation function).

2 Related Work

Activation functions play a critical aspect in the success of deep learning [7]. They provide the necessary fitting ability to neural networks and deep neural networks to learn complex patterns since the activation function will determine the functional subspace that the optimization algorithm will explore. However, it is still unclear why one function performs better than another one. In [37] summarize certain heuristics currently known to judge which may perform better than another one presented in table 1. *Hayou* et al. [14] shows that smooth activation functions provide better signal propagation through the network. [36] and [32] show that the output landscape of the network has a deep relation with the optimization process of this network. Unboundedness is desirable because it avoids saturation, where training is slow due to near-zero gradients. Furthermore, smoothness plays a beneficial role in optimization and generalization [9, 32, 36]. Functions that approach zero in the limit induce even larger regularization effects because large negative inputs are easily ”forgotten”. Finally, the non-monotonicity of an activation function increases its expressivity and improves gradient flow.

Ding et al. [7] experiments with deep architectures and the status and the developments of commonly used activation functions: Sigmoid, Tanh, ReLU [33], and ELU [5]. Based on the complexity of experiments we do not consider [7] as a deep study.

The activation functions used in deep neural networks have a significant impact on the training, dynamics, and performance of any neural network [36]. In [36] a neural architecture search algorithm is used to generate a new activation function.

Villmann et al. [48] handle specifically the problem of activation function for Multi-Layer Perceptron (MLP) [40]. They concluded that Softplus [8], the parameterized sigmoid and Swish [36] perform better than ReLU [33] for MLP [40].

Pedamonti et al. [35] compared 3 activation functions (ReLU, Leaky ReLU and SELU) on the MNIST dataset. They concluded that the learning process is faster with ELU and SELU compared to ReLU and Leaky ReLU.

Ohn et al. [34] introduce the usage of a general activation function. They established the upper bounds of the required depth, width, and sparsity of deep neural networks to approximate any Hölder continuous function for a general class of activation functions. The rationale behind their works is that piece-wise linear activation functions are more efficient in approximating local basis functions while locally quadratic activation functions are more efficient in approximating polynomials.

Gerard Timmons et al. [47] shows that the most common used activation function, ReLU is not always the best solution for Recurrent Neural Network (RNN). They proposed an approximation of sigmoid and tanh to further reduce the inference and training time of neural networks.

Table 1: Frequent properties of activation functions [37].

Property	Description	Problems	Examples
derivative	f'	> 1 exploding gradient (e) < 1 vanishing (v)	Sigmoid, Hyperbolic Tangent
zero-centered	range centered around zero?	if not, slower learning	Hyperbolic Tangent
saturating	finite limits	vanishing gradient in the limit	Hyperbolic Tangent, ReLU
monotonicity	$x > y \implies f(x) \geq f(y)$	unclear	exceptions: Swish

3 Activation Function

In this section, we describe the activation functions used and compared them during our experimentation.

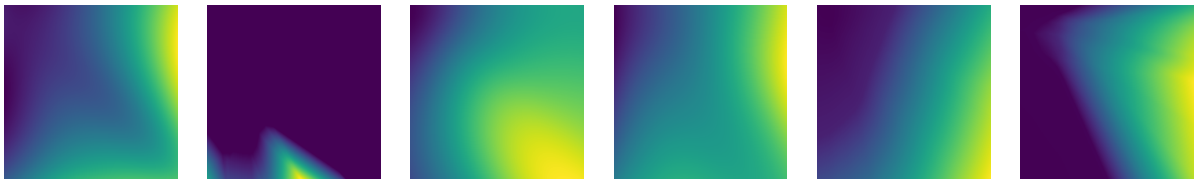
3.1 Hytana

We propose a new activation function: Hytana. Hytana is a self-gated smooth function, unbounded below, non-monotonic, and has a global minimum of approximately 0.344 at around $x \approx -1.136$. The concave part of the function allows the network to preserve small negative input and therefore increase its representational sparsity. In section 3.3 we explain how we chose the value 0.303 and 0.632. This function is defined as follows:

$$h(x) = \frac{x + x\tanh(\theta)}{2} \quad \text{where } \theta = 0.303 + 0.632x \tag{1}$$

$$\frac{d}{dx}h(x) = \frac{1 + 0.632x\,\text{sech}^2(\theta) + \tanh(\theta)}{2} \quad \text{where } \theta = 0.303 + 0.632x \tag{2}$$

The output landscape of Hytana is shown in figure 1a and figure 2 shows the plot of Hytana, its 1st, and 2nd derivative. This output landscape has been computed from a 4 linear layers neural network initialized with Kaiming Uniform [16] using Hytana as an activation function after every layer including the last one.



(a) Hytana. (b) Relu. (c) Mish. (d) Swish. (e) Prelu. (f) LeakyRelu.

Figure 1: Output landscape of Hytana, Relu, Swish, Mish, Prelu, LeakyRelu from a 4 layers neural network initialized with Kaiming Uniform [16]. The smoothness of this landscape results in smoother loss functions which are easier to optimize [32, 36].

3.2 Parametric Hytana (PHytana)

We show that replacing the two constant parameters in Hytana with two learnable parameters improves accuracy and convergence. Those parameters are first initialized as the same value as Hytana (eq. 1). Therefore when $\alpha = 0.303$ and $\beta = 0.632$ PHytana becomes Hytana.

$$p(x) = \frac{x + x\tanh(\alpha + \beta x)}{2} \tag{3}$$

where α and β are learnable parameter initialized at $\alpha = 0.303$ and $\beta = 0.632$.

$$\frac{d}{dx}p(x) = \frac{1 + \beta x\,\text{sech}^2(\alpha + \beta x) + \tanh(\alpha + \beta x)}{2} \tag{4}$$

Multiple modern activation functions have that concave part on the left side. The parameterization of those two parameters controls the length and height of this concave part or completely removes it. PHytana can be trained using backpropagation [27] and optimized simultaneously with other layers.

PHytana can be used in two fashions: with global α and β parameters, and with a channel-wise variant where each channel has its own α and β. Therefore it introduces a small number of extra parameters. For the first variant, the extra number of parameters is equal to the number of layers by 2. For the channel-wise variant, the number of extra parameters is equal to the total number of channels by 4. In both cases, the extra number of parameters is negligible when considering the total number of weights. So we expect no extra risk of overfitting.

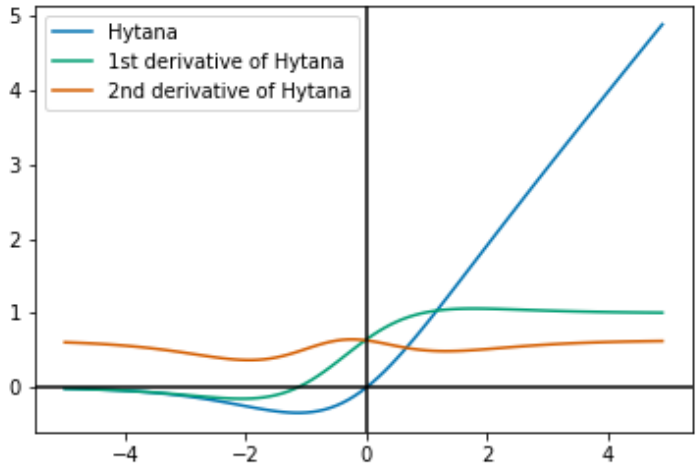


Figure 2: Plot of Hytana and its derivative.

To illustrate the channel-wise variant of PHytana, we replace α and β from the equation 3 with the vector $\boldsymbol{A}$ and $\boldsymbol{B}$ containing each parameter for the channel. Hence the equation becomes:

$$p(x) = \frac{x + x\tanh(\boldsymbol{A} + \boldsymbol{B}x)}{2} \tag{5}$$

In this study, we only focus only on the first variant.

3.3 Motivation

Swish [36] and Mish [32] both improve upon Relu. However, despite their similarity, Mish is significantly better than Swish. This small difference induces around $\approx 1\%$ increase in accuracy across multiple datasets. The idea behind this study is first to find a function that has the same behavior as Mish. But at the same time have enough variables to easily manipulate the width and the height of the concave part (Hytana).

Secondly, let those variables be optimized by gradient descent (PHytana). As a starting point, we choose a function in the form of $\frac{x(1+\tanh(\alpha+\beta x)}{2}$. We did a grid search to find good values for α and β to obtain good accuracy. We stumble upon a couple of possible candidates $\alpha = 0.684, \beta = 1$ and $\alpha = 0.303, \beta = 0.632$. After further experimentation, we discovered that the latter provides consistent results. We unfailingly use those values for all our following experiences.

Furthermore, Phytana can approximate Swish, Mish or other activations function with a certain chosen value for α and β. For example with a value of $\alpha = 0.65940366$ and $\beta = 0.31554185$, our function approximate ELU [5] or with $\alpha = -0.00599371$ and $\beta = 0.99990216$ for ReLU [9]. We call this behavior self-adaptability. As shown in figure 3, this allows the network a wide range of expressivity due to the neural network; and the values of α and β may change in every layer.

4 Datasets

4.1 Cifar-10 and Cifar-100

Cifar-10 and Cifar-100 were first introduced in [25]. They are currently the most used datasets in the field of artificial intelligence. Cifar-10 consists of 60k of 32x32 color images and contains 10 classes. The Cifar-100 is an extension of CIFAR-10. It has 100 classes with 600 images for each class. There are 500 training images and 100

Table 2: Summary of Activation function

NAME	EQUATION	DERIVATIVE
SIGMOID	$\frac{1}{1+e^{-x}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$
HYPERBOLIC TANGENT	$\frac{2}{1+e^{-2x}} - 1$	$\frac{1}{\cosh^2 x}$
RELU [16] [9]	$\max(0, x)$	$\begin{cases} 0, & \text{IF } x \leq 0 \\ 1, & \text{IF } x > 0 \end{cases}$
LEAKY RELU [16]	$\max(0, x) + \alpha \min(0, x)$	$\begin{cases} 1, & \text{IF } x \geq 0 \\ \alpha, & \text{OTHERWISE} \end{cases}$
PRELU [16]	$\max(0, x) + \alpha \min(0, x)$	$\begin{cases} 1, & \text{IF } x \geq 0 \\ \alpha, & \text{OTHERWISE} \end{cases}$
RELU6 [2]	$\min(\max(0, x), 6)$	$\begin{cases} 0, & \text{IF } x > 6 \vee x < 0 \\ 1, & \text{IF } 6 \geq x \geq 0 \end{cases}$
ELU [5]	$\max(0, x) + \min(0, \alpha(e^x - 1))$	$\begin{cases} \alpha e^x, & \text{IF } x < 0 \\ 1, & \text{IF } x \geq 0 \end{cases}$
SELU [24] [1]	$\gamma(\max(0, x) + \min(0, \alpha(e^x - 1)))$	$\gamma \begin{cases} \alpha e^x, & \text{IF } x < 0 \\ 1, & \text{IF } x \geq 0 \end{cases}$
SWISH [36]	$\frac{x}{1+e^{-x}}$	$\frac{e^x(x+e^x+1)}{(e^x+1)^2}$
MISH [32]	$x \tanh(\log(1 + e^x))$	$\frac{e^x(4e^x x+4x+6e^x+4e^{2x}+e^{3x}+4)}{(2e^x+e^{2x}+2)^2}$
HARDSHRINK	$\begin{cases} x, & \text{IF } x > \lambda \\ -x, & \text{IF } x < -\lambda \\ 0, & \text{OTHERWISE} \end{cases}$	$\begin{cases} 1, & \text{IF } x > \lambda \\ -1, & \text{IF } x < -\lambda \\ 0, & \text{OTHERWISE} \end{cases}$
HARDTANH	$\begin{cases} 1 & \text{IF } x > 1 \\ -1 & \text{IF } x < -1 \\ x & \text{OTHERWISE} \end{cases}$	$\begin{cases} 0, & \text{IF } x > 1 \\ 0, & \text{IF } x < -1 \\ 1, & \text{OTHERWISE} \end{cases}$
RRELU [50]	$\begin{cases} x & \text{IF } x \geq 0 \\ ax & \text{OTHERWISE} \end{cases}$	$\begin{cases} 1, & \text{IF } x \geq 0 \\ a, & \text{OTHERWISE} \end{cases}$
CELU [3]	$\begin{cases} x & \text{IF } x \geq 0 \\ a(e(x) - 1) & \text{OTHERWISE} \end{cases}$	$\begin{cases} 1, & \text{IF } x \geq 0 \\ a(e(x)), & \text{OTHERWISE} \end{cases}$
SOFTPLUS	$\frac{1}{\beta} * \log(1 + e^{\beta * x})$	$\frac{e^{\beta x}}{e^{\beta x}+1}$
TANHSHRINK	$x - \tanh(x)$	$1 - \text{sech}^2(x)$
HARDSIGMOID	$\frac{1}{6} \min(\max(0, x + 3), 6)$	$\begin{cases} \frac{1}{6} & \text{IF } -3 < x < 3 \\ 0 & \text{OTHERWISE} \end{cases}$
HARDSWISH	$\frac{1}{6} x \min(\max(0, x + 3), 6)$	$\begin{cases} 1 & \text{IF } x > 3 \\ \frac{1}{6}(2x + 3) & \text{IF } -3 < x < 3 \\ 0 & \text{IF } x < -3 \text{ OTHERWISE} \end{cases}$

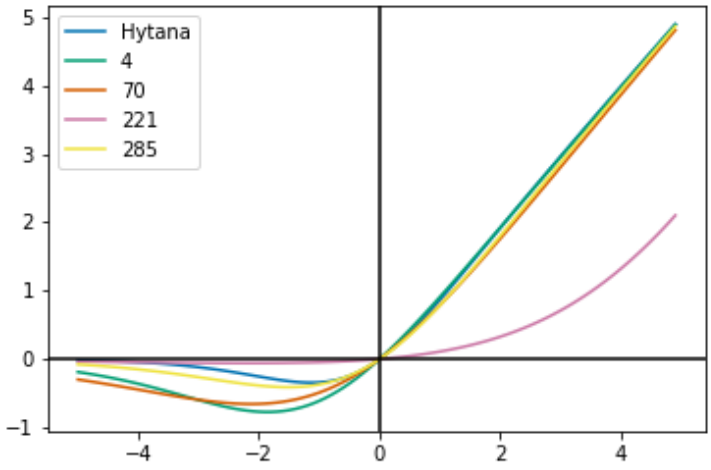


Figure 3: Parameters α and β found by the network for different layer with PHytana on CIFAR-10. The legend represents the index of the layer in EfficientNet-B0. We see how the network controls the concave portion of the activation function from the earlier layer to the later one.

testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a ”fine” label (the class to which it belongs) and a ”coarse” label (the superclass to which it belongs).

4.2 Caltech-256

The Caltech 256 [11] is considered an improvement of the Caltech 101 dataset. It consists of 256 object types and together 30 607 RGB images represent a wide variety of artificial or natural objects in different conditions. For each object, there are at least 80 images. Images are ready to use after downloading the dataset, without the need for pre-processing. Images were manually chosen from two popular online image databases. This dataset has different lighting conditions, poses, backgrounds, image sizes, and camera systematics.

5 Models

5.1 ResNet

ResNet was first introduced in [17] and achieved state-of-the-art accuracy within the ImageNet [6] challenge. ResNet is mainly characterized by the skip connections between layers as it’s shown in figure 4. This architecture is able to learn and converge faster than a shallower Neural Network not using skip connections. Experimentation with ImageNet, MNIST, Cifar-10, and Cifar-100 showed that ResNet outperformed traditional neural networks such as LeNet-5 [27], AlexNet [26], FitNet [39], VGGNet [42] and Highway [43].

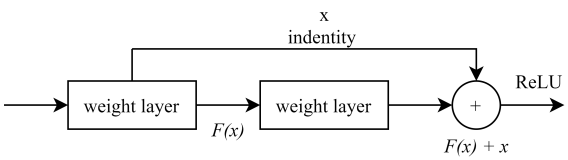


Figure 4: Residual learning: a building block [17]

In *Why ResNet Works? Residuals Generalize* [15] shows that residual connections are part of state-of-the-art neural network architectures. They can improve the performance of deep neural networks. A generalization bound for ResNet was proposed, *i.e*. ResNet will generalize better [15].

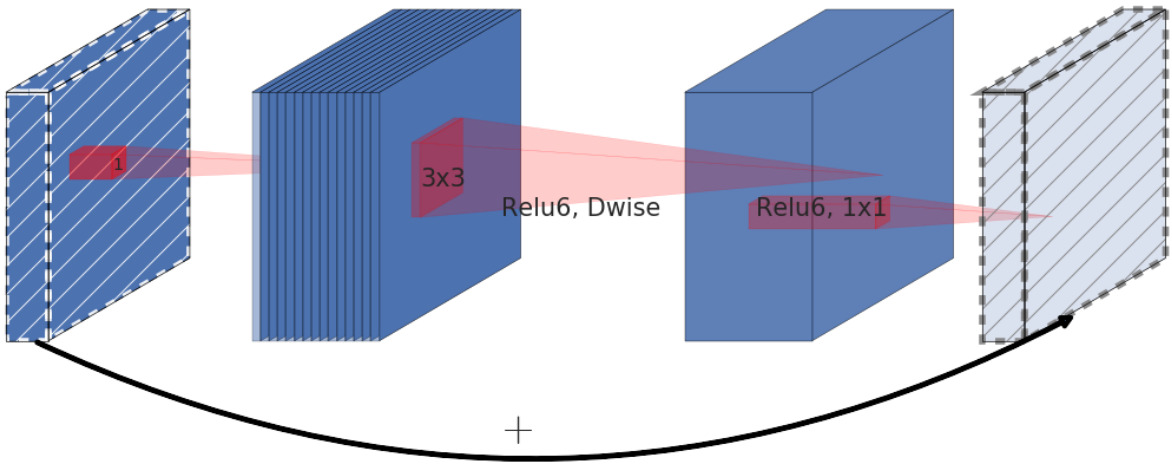


Figure 5: Illustrate the Inverted Residual Block as used in EfficientNet and MobileNet. Inside an Inverted Residual Block, skip connections connect narrow layers while wider layers are between skip connections.

5.2 ResNet-D

ResNet-D changes the downsampling block of a regular ResNet and adds 2×2 average pooling layer with a stride of 2 before the convolution, whose stride is changed to 1. This architecture has been first introduced in Torch [12] and later in other works by [1], [21], [49] and finally formalized by [18]. Figure 6 illustrates the change made in the residual block of ResNet-D.

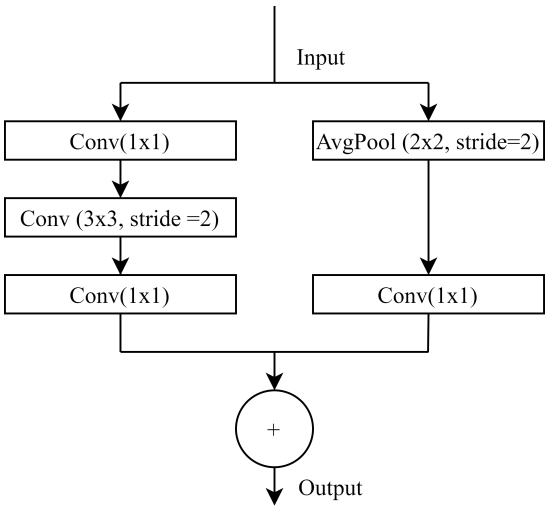


Figure 6: A Residual block of ResNet-D.

5.3 EfficientNet

EfficientNet was first introduced in [45]. They used neural architecture search to design a new architecture and scale it to obtain a family of models (EfficientNet B0 to B8). The discovered architecture achieves state-of-the-art classification accuracy on ImageNet while being 8.4x smaller and 6.1x faster on inference than the best existing Convolutional Neural Network. It achieves a high level of accuracy on many other datasets like CIFAR-100, Flowers, and Cars. EfficientNet makes heavy use of inverted residual block (figure 5), depthwise convolution, pointwise convolution, and Squeeze and excitation block [21]. In our experiment we used EfficientNet-B0.

5.4 MobileNet

MobileNet was first introduced in [20] as an efficient deep learning model for mobile and embedded applications. This architecture is mainly based on depthwise separable convolutions. MobileNet v2 [41] is an improvement upon MobileNet by introducing the usage of inverted residual blocks as shown in figure 5. Finally, MobileNetv3 [19] used platform-aware Neural Architecture search algorithms mainly NetAdapt [51] to find new architecture for mobile and embedded platforms. [19] proposed two variants of MobileNetV3: MobileNetV3-Large and MobileNetV3-small for high and low resource use cases respectively. In our experiment we used MobileNetV3-Large.

6 Experimentation

In this section, we describe our result but beforehand we describe the environment of our experiment and some implementation details.

6.1 Experiment Setup

All experiments were done on GPU NVidia RTX 2070 super. As an optimizer, we used Adam [23], with a learning rate of 0.001, batch size 512, and no data augmentation. As a criterion, we used cross-entropy. Each dataset has been normalized. For Cifar-10, and Cifar-100 the provided train/test set was used. For Caltech-256 we used split the data 80% for training and 20% for evaluation. We employed the same split of the training and evaluation set for every model. All models have been trained 5 spans and the measure shown in this study is the mean of those 5 iterations if not mentioned otherwise. Kaiming He initialization [16] was used for the initialization of every layer if not mentioned otherwise. No form of learning rate annealing was used. The expanded results are visualized in the Appendix.

6.2 Implementation Details

We changed the activation function in every layer according to the architecture. The initialization method used for Resnet-D is the same as in [18] except for every last Batch Normalization (BN) layer inside the residual block which was initialized following the zero γ initialization heuristic [10]. For MobileNetV3-Large and EfficientNet, we used the same initialization method as in [44].

6.3 Results and Discussion

We did a performance comparison of those activation functions. We performed a test with 10 000 runs with a warmup of 10 runs on the GPU for every activation function on 16-bit floating-point precision (FP16) and 32-bit floating-point precision (FP32).

On FP16, Leaky ReLU is the fastest followed by ReLU and Hardshrink with only a difference of $0.1\mu s$. However, the difference between Swish ($204.2\mu s$), Mish ($245.8\mu s$), and Hytana ($210.5\mu s$) to ReLU is in the order of $\pm 50\mu s$ hence negligible. Table 3 resume the performance of those activation function on FP16. A full table will be presented in the appendix for FP16 and FP32.

Table 8 and table 9 summarize the performance on CIFAR 10. The Table 4, 7 shows the performance on Caltech-256.

Table 3: Summary of performance comparison of activation function on FP16.

ACTIVATION FUNCTION	MEAN ± STD
LEAKY RELU FWD	$161.0\mu s \pm 23.46\mu s$
LEAKY RELU BWD	$1.297ms \pm 1.384ms$
RELU FWD	$163.1\mu s \pm 8.497\mu s$
RELU BWD	$1.371ms \pm 1.399ms$
SWISH FWD	$204.2\mu s \pm 41.03\mu s$
SWISH BWD	$1.347ms \pm 1.375ms$
MISH CUDA FWD	$245.8\mu s \pm 241.3\mu s$
MISH CUDA BWD	$1.398ms \pm 1.443ms$
HYTANA FWD	$210.5\mu s \pm 33.94\mu s$
HYTANA BWD	$1.449ms \pm 1.345ms$

Table 5: Train and test loss on Cifar-10/100 dataset with ResNet-D on all activation function after 100 epochs

Act. function	Phytana	Hytana	Mish	Prelu	Swish	Hswish	Lrelu	Relu	Relu6	Rrelu	Elu
Train loss cifar-10	0.030	0.032	0.033	0.034	0.034	0.037	0.041	0.041	0.042	0.058	0.059
Test loss cifar-10	0.677	0.665	0.670	0.721	0.678	0.687	0.709	0.710	0.713	0.674	0.612
Train loss cifar-100	0.044	0.046	0.051	0.050	0.051	0.053	0.067	0.065	0.066	0.102	0.081
Test loss cifar-100	0.677	0.665	0.670	0.721	0.678	0.687	0.709	0.710	0.713	0.674	1.899

Act. function	Celu	Htanh	Softplus	Tanhshrink	Selu	Softsign	Sigmoid	Hsigmoid	Hshrink
Train loss cifar-10	0.060	0.071	0.078	0.081	0.105	0.109	0.234	0.442	0.641
Test loss cifar-10	0.618	0.761	0.726	0.771	0.629	0.738	2.373	2.442	0.814
Train loss cifar-100	0.081	0.101	0.104	0.097	0.132	0.182	0.219	0.832	1.696
Test loss cifar-100	1.909	2.193	2.250	2.433	1.918	2.149	4.485	4.610	2.297

Table 4: Accuracy on the training set of Caltech-256 after 10 and 20 epochs with ResNet18.

	ACCURACY	
	10 EPOCHS	20 EPOCHS
1	PHYTANA (77.88%)	PHYTANA (99.98%)
2	SWISH (74.79%)	HARDSWISH (99.94%)
3	HARDSWISH (74.55%)	MISH (99.92%)
4	RELU6 (70.52%)	LEAKY RELU (99.91%)
5	LEAKY RELU (69.54%)	SWISH (99.91%)
...	...	...
20	HARDSIGMOID (14.75%)	HARDSIGMOID (26.03%)

Table 4 points out that PHytana had the highest performance on the training set of Caltech-256 after 10 and likewise after 20 epochs. Our experiments with Caltech-256 did not involve EfficientNet-b0 and MobileNetV3-Large architecture. The training accuracy after 10 and 20 epochs is shown because the neural networks start overfitting later and the difference in performance is in the order of $10 - 3$ in loss and $10 - 6$ in gain.

Table 6 and 5 show the obtained loss and accuracy at the end of the training (100 epochs). In the long run, Hytana and Phytana outperformed most of the other activation functions. Swish and Mish remain very close. However, we argue that the self-adaptability of PHytana gives it the upper hand.

Table 6: Train and test Accuracy on Cifar-10/100 with ResNet-D on all activation functions after 100 epochs.

Act. functions	Phytana	Hytana	Mish	Prelu	Swish	Hswish	Lrelu	Relu	Relu6	Rrelu	Elu
Train acc. cifar-10	98.97	98.88	98.86	98.83	98.81	98.71	98.60	98.58	98.56	97.97	97.92
Test acc. cifar-10	79.94	80.62	80.54	79.01	80.61	80.42	79.42	79.64	79.64	80.29	82.10
Train acc. cifar-100	98.61	98.51	98.36	98.42	98.38	98.29	97.83	97.91	97.86	96.64	97.39
Test acc. cifar-100	48.38	49.53	49.04	46.12	48.89	48.57	47.42	47.79	47.48	50.36	53.06

Act. functions	Celu	Htanh	Tanhshrink	Softplus	Selu	Softsign	Sigmoid	Hsigmoid	Hshrink
Train acc. cifar-10	97.88	97.51	97.24	97.21	96.24	96.15	91.59	84.22	77.48
Test acc. cifar-10	82.01	76.73	77.05	80.61	81.73	77.47	32.36	18.34	72.48
Train acc. cifar-100	97.37	96.85	96.94	96.57	95.64	94.29	93.42	75.18	53.49
Test acc. cifar-100	52.96	45.71	43.25	50.38	53.18	46.50	5.77	2.33	42.62

Table 7: Accuracy on the test set of Caltech-256 after 10 and 20 epochs with ResNet18.

	ACCURACY	
	10 EPOCHS	20 EPOCHS
1	MISH (45.52%)	MISH (50.14%)
2	HARDSWISH (44.88%)	PHYTANA (49.11%)
3	PHYTANA (44.86%)	HARDSWISH (48.93%)
4	HYTANA (44.85%)	LEAKY RELU (48.27%)
5	SWISH (44.03%)	HYTANA(48.01%)
...	...	...
20	HARDSIGMOID (5.9%)	HARDSIGMOID (17.57%)

The results obtained from the test set are presented in the table 7. The accuracy is lower than the metrics on the training set. The performance difference between the test set and the train set is probably due to the absence of data augmentation and the class distribution on the training and test set.

Table 8: Accuracy on the train set of Cifar-10 after 10 and 20 epochs with ResNet18.

	ACCURACY	
	10 EPOCHS	20 EPOCHS
1	PHYTANA (81.18%)	PHYTANA (90.61%)
2	PRELU (80.81%)	PRELU (89.57%)
3	HYTANA (80.13%)	MISH (88.19%)
4	MISH (79.5%)	SWISH (87.76%)
5	SWISH (79.46%)	HYTANA (87.6%)
...	...	...
20	TANHSHRINK (27.25%)	TANHSHRINK (59.11%)

The results in table 8 show that after 10 epochs there is a big gap in performance between the best and the worst activation function. PHytana progressed almost 10% within 10 extra epochs.

Table 9: Accuracy on the test set of Cifar-10 after 10 and 20 epochs with ResNet18.

	ACCURACY	
	10 EPOCHS	20 EPOCHS
1	HYTANA (77.1%)	HYTANA (79.91%)
2	PHYTANA (76.98%)	ELU (79.41%)
3	SWISH (76.39%)	SWISH (78.89%)
4	HARDSWISH (75.89%)	HARDSWISH (78.86%)
5	ELU (75.22%)	CELU (78.77%)
...	...	...
20	SIGMOID (10.02%)	SIGMOID (29.51%)

Training accuracy in table 9 shows that after 10 epochs Hytana and PHytana took first and second place. Contrariwise we can say that sigmoid is close to a random guess after 10 epochs in the training set.

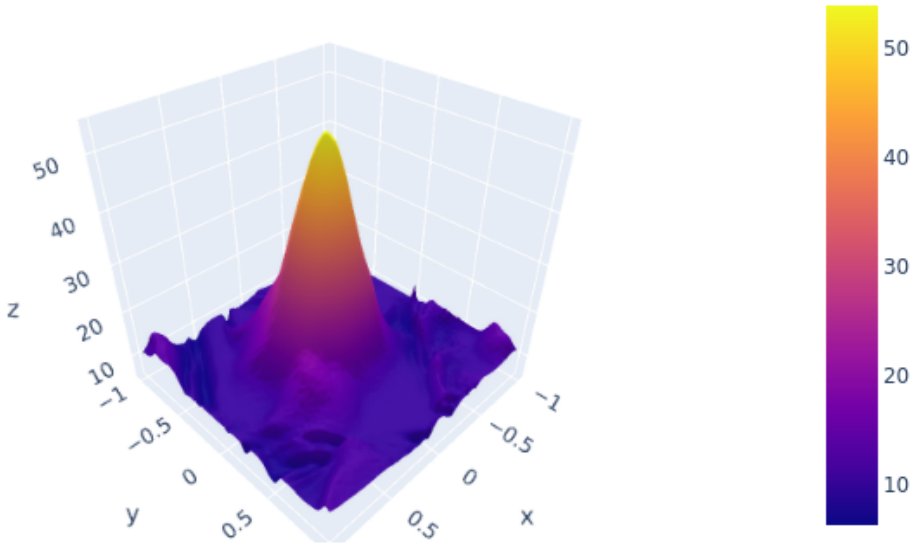


Figure 7: The accuracy landscape of ResNet-D using PHytana as activation function after one epoch in Cifar-10. We see how smooth the surface is. Hence it facilitates the optimization process.

The loss and accuracy landscape was visualized following the method mentioned in [29]. The landscape represented in figure 7 shows the accuracy of ResNet-D after one epoch in Cifar-10. The x-axis and y-axis represent weights of the neural network, where n-dimensional space has been reduced to 2-dimensional space.

Rectifier networks are easier to train [9, 26, 52] compared to traditional sigmoid-like activation networks. But surprisingly enough Sigmoid and Softplus surpassed Swish on EfficientNet-B0 on Cifar-10 after 90 epochs. Among all tested activation functions HardShrink, TanShrink and HardTanH performed the worst on all 3 datasets and 4

architectures.

On Cifar-100 using EfficientNet, Hytana was ranked fifth after performing 20 and 100 epochs. However, PHytana was in first place after 20 epochs and after 100 epochs in second place.

With Resnet18, Hytana was in third place in both cases and Phytana was in second place after PReLU after 20 epochs and first after 100 epochs. With Resnet-D 18 which is just a modification of the previous model. Hytana was in third place after 20 epochs and in second place after 100 epochs, Phytana did the same performance as in Resnet18. On the last model MobileNet-v3, Hytana was ranked fourth in both cases. However, PHytana ends up in first place in both cases. During the training, both functions did a perfect performance, which means that they outperform most of the activation functions. After 20 epochs Tanhshrink activation function did the worst performance except with ResNet18 where Hardshrink did the worst performance.

On EfficientNet, Tanshrink activation function had an accuracy of 1.11%, which is pretty close to a random guess while having 100 classes in Cifar-100. However, PHytana's accuracy was 51.85% after 20 epochs and 96.32% after 100 epochs.

After 100 epochs (Table 6) it is just the reverse situation. *I.e*. only one case where Hardshrink was not the worst. It was on EfficientNet where Tanshrink had the worst performance. In the testing part, we need to give some insights from our experiment:

- The aim of this work was not to achieve the highest accuracy after 100 epochs, but to reach the best accuracy in the shortest amount of iteration possible;
- Hytana and PHytana have always converged to the local minimum between the 10th and 25th epochs;
- PHytana converged faster than the most activation function, however, given enough time, Mish, PReLu, Elu, and Swish were close to that performance;
- When PHytana gets outperformed most of the time, the difference is in the order of 10^{-2}.
- Across every run: Hardsigmoid, Hardshrink, and Sigmoid always provide the worst result.

7 Conclusion

Activation functions have a large influence on the overall performance of neural networks, therefore finding the optimum parameters can be difficult and it is not even clear whether we reach the optimum parameters or not. As shown in our experiment result, picking the wrong activation function can have a crucial impact. In this work, we compared the 18 most used activation functions using 4 modern neural network architectures applied to 3 different datasets. The result of our experiment permitted us to introduce two new activation functions namely Hytana and PHytana. Through the mean of extensive testing and benchmarking, we showed that PHytana can converge faster than most of the other activation functions and also shows competitive performance in the long run. We further showed that the time accuracy tradeoff doesn't hold as the difference between ReLU and PHytana is in the order $\pm 50\mu s$. We leave for future work the study of the channel-wise variant of PHytana and the application of those activation functions on other types of networks such as Recurrent Neural Networks. Although the number of possible activation is limitless, our research tries to fill the lack of knowledge in this area by testing the most plausible one.

References

[1] Yousef Al-Qudah and Nasruddin Hassan. Operations on complex multi-fuzzy sets. *Journal of Intelligent & Fuzzy Systems*, 33(3):1527–1540, 2017.

[2] Alex Krizhevsky. Convolutional Deep Belief Networks on CIFAR-10.

[3] Jonathan T Barron. Continuously differentiable exponential linear units. *arXiv preprint arXiv:1704.07483*, 2017.

[4] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv:1511.07289 [cs]*, February 2016. arXiv: 1511.07289.

[6] 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.

[7] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. *2018 Chinese Control And Decision Conference (CCDC)*, 06 2018.

[8] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *Advances in neural information processing systems*, pages 472–478, 2001.

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, June 2011.

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. June 2017.

[11] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[12] Sam Gross and Michael Wilber. Training and investigating residual nets. *Facebook AI Research*, 6, 2016.

[13] Matthew Guzdial and Mark Riedl. Automated game design via conceptual expansion. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2018.

[14] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the Impact of the Activation Function on Deep Neural Networks Training. *arXiv:1902.06853 [cs, stat]*, May 2019. arXiv: 1902.06853.

[15] Fengxiang He, Tongliang Liu, and Dacheng Tao. Why resnet works? residuals generalize. *arXiv preprint arXiv:1904.01367*, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*, February 2015. arXiv: 1502.01852.

[17] 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.

[18] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.

[20] 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.

[21] 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.

[22] Yueming Jin, Huaxia Li, Qi Dou, Hao Chen, Jing Qin, Chi-Wing Fu, and Pheng-Ann Heng. Multi-task recurrent convolutional network with correlation loss for surgical video analysis. *Medical image analysis*, 59:101572, 2020.

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

[24] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. *arXiv:1706.02515 [cs, stat]*, September 2017. arXiv: 1706.02515.

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

[26] 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.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Chenge Li, Gregory Dobler, Xin Feng, and Yao Wang. Tracknet: Simultaneous object detection and tracking and its application in traffic video analysis. *arXiv preprint arXiv:1902.01466*, 2019.

[29] 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.

[30] Xia Liang, Junmin Wu, and Jing Cao. Midi-sandwich2: Rnn-based hierarchical multi-modal fusion generation vae networks for multi-track symbolic music generation. *arXiv preprint arXiv:1909.03522*, 2019.

[31] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples, 2019.

[32] Diganta Misra. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *arXiv:1908.08681 [cs, stat]*, October 2019. arXiv: 1908.08681.

[33] 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.

[34] Ilsang Ohn and Yongdai Kim. Smooth Function Approximation by Deep Neural Networks with General Activation Functions. *Entropy*, 21(7):627, June 2019.

[35] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on mnist classification task, 2018.

[36] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for Activation Functions. *arXiv:1710.05941 [cs]*, October 2017. arXiv: 1710.05941.

[37] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčak. A review of activation function for artificial neural network. In *SAMI 2020 • IEEE 18th World Symposium on Applied Machine Intelligence and Informatics*, 01 2020.

[38] Abhianv Ravi, Arun Patro, Vikram Garg, Anoop Kolar Rajagopal, Aruna Rajan, and Rajdeep Hazra Banerjee. Teaching dnns to design fast fashion. *arXiv preprint arXiv:1906.12159*, 2019.

[39] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[40] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[41] Mark Sandler, 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*, pages 4510–4520, 2018.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[44] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

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

[46] Ivona Tautkute, Tomasz Trzciński, Aleksander P Skorupa, Łukasz Brocki, and Krzysztof Marasek. Deepstyle: Multimodal search engine for fashion and interior design. *IEEE Access*, 7:84613–84628, 2019.

[47] Nicholas Gerard Timmons and Andrew Rice. Approximating activation functions. *arXiv preprint arXiv:2001.06370*, 2020.

[48] Thomas Villmann, John Ravichandran, Andrea Villmann, David Nebel, and Marika Kaden. Activation functions for generalized learning vector quantization-a performance comparison. *arXiv preprint arXiv:1901.05995*, 2019.

[49] 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.

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

[51] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[52] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On rectified linear units for speech processing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521. IEEE, 2013.

[53] Alexander Zook and Mark O Riedl. Automatic game design via mechanic generation. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.