World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:17, No:11, 2023

An Empirical Study on Switching Activation Functions in Shallow and Deep Neural Networks

Apoorva Vinod, Archana Mathur, Snehanshu Saha

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

Abstract—Though there exists a plethora of Activation Functions (AFs) used in single and multiple hidden layer Neural Networks (NN), their behavior always raised curiosity, whether used in combination or singly. The popular AFs – Sigmoid, ReLU, and Tanh – have performed prominently well for shallow and deep architectures. Most of the time, AFs are used singly in multi-layered NN, and, to the best of our knowledge, their performance is never studied and analyzed deeply when used in combination. In this manuscript, we experiment on multi-layered NN architecture (both on shallow and deep architectures; Convolutional NN and VGG16) and investigate how well the network responds to using two different AFs (Sigmoid-Tanh, Tanh-ReLU, ReLU-Sigmoid) used alternately against a traditional, single (Sigmoid-Sigmoid, Tanh-Tanh, ReLU-ReLU) combination. Our results show that on using two different AFs, the network achieves better accuracy, substantially lower loss, and faster convergence on 4 computer vision (CV) and 15 Non-CV (NCV) datasets. When using different AFs, not only was the accuracy greater by 6-7%, but we also accomplished convergence twice as fast. We present a case study to investigate the probability of networks suffering vanishing and exploding gradients when using two different AFs. Additionally, we theoretically showed that a composition of two or more AFs satisfies Universal Approximation Theorem (UAT).

Keywords—Activation Function, Universal Approximation function, Neural Networks, convergence.

I. Introduction

IN the past few decades, owing to a rise in computing power, Artificial Neural Networks (ANNs) have found wide-ranging applications in a variety of fields. ANNs are usually made up of components called layers, consisting of neurons. ANNs have input layers, hidden layers, and a final output layer. Information from the previous layer is sent to the next layer, where each input to the next layer is multiplied with its relevant weights and a bias is added. AFs are then employed on this weighted sum, to deliver an output to the next layer. These AF are generally used to impart non-linearity to a neuron, in order to capture a wider range of functions that an output must take. They also have a significant impact on the speed at which the neural network converges to a minima, and help in normalising the output.

Various AFs like Sigmoid, ReLU and Tanh have been used in neural networks. Variants like leaky ReLU and PReLU have also been used to combat issues that ReLU has; i.e. vanishing gradients due to too many negative values. Sigmoid has a vanishing gradient issue as well, due to the low magnitude of the gradients of the sigmoid function. To combat these issues, many optimizers have been used instead of the traditional gradient descent. SGD with momentum uses the first moment, a running average of previous gradients, which speeds up convergence and helps avoid local minima traps. RMSProp instead normalizes gradient by its root mean square (as opposed to just previous gradients) and ensures that gradients for each weight have unique learning rates. Adam combines both momentum and RMSProp and results in tweakable hyperparameters. Adam is the optimizer that is most widely used.

Activation Functions (AF) are building blocks for performing predictive tasks in Neural Networks. Generally, the same AF in different layers is used, be it Sigmoid, ReLU, or Tanh. Extensive empirical studies on the behavior and effectiveness of different AFs have been conducted on different data sets and in shallow and deep networks, with narrow and wide architectures. The application of the neural network is not just limited to the field of classification and regression, but also to model control systems in power plants like de-superheater for controlling temperature [1]. Both linear and non-linear AFs are used in modeling the de-superheater. Likewise, Jianli et al. [2] and Sibi et al. [3] have studied the behavior of AF ranging from all variants of ReLU, sigmoid, tanh, Elliot, Swish, EliSH, and Hexpo [4], [5] on shallow as well as deep neural architectures but lacking any robust empirical or experimental analysis for their work. Saha et al. [6] studied and proposed new AFs in the parabolic form by using a Diffact-based framework. Another set of research experiments to determine whether two hidden layers are optimal or one was conducted by Hornik et al. [7]. Makhoul et al. [8] explored how a two-layered NN separates the input space into distinct decision regions on the first and the second layer. While Brightwell et al. [9] analyzed the benefits of having one hidden layer over two, Wan et al. [10] proposed an enhanced back-propagation algorithm for updating selective weights on particular epochs to ensure a noise-insensitive training and a better classification. Most of these research findings dealt with the expressive power of hidden layers associated with the hidden units in each layer. They lacked the exploration of different AFs used in combination, within the same network, and across hidden layers with the perspective of achieving cross-pollination of AFs facilitating significant network performance and better classification metrics.

A. Motivations

The idea to leverage two different AFs alternately (in the sense of composition of affine transformations) in the

Apoorva Vinod and Archana Mathur are with the Department of Information Science and Engineering, Nitte Meenakshi Institute of Technology, Yelahanka, Bangalore, India (mathurarchana77@gmail.com).

Snehanshu Saha is with the Department of Computer Science and Information Systems and APPCAIR, BITS Pilani K K Birla Goa Campus (e-mail: scibase.snehanshu@gmail.com).

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

hidden layers instead of using a common AF is thus worth investigating. Subsequently, an attempt to study the effects of using two different AFs in subsequent layers coupled with a mathematical justification of the approximation ability of such composition is reasonably rational. However, such claims need to be supported by empirical results which might validate the hypothesis that a neural network using this kind of combination of two AFs could result in better outcomes when compared to a neural network using the same AF for both layers. The paper demonstrates how the AFs are employed in the network to generate good classification results and investigates the link between various AFs when applied to the hidden and output layers. The goal is to create a Neural network capable of performing the following tasks: to replicate the behavior of all AFs on the hidden and output layers and also observe the operation of the commutative property of the AF (Sigmoid, Tanh, and ReLU) on the hidden and output layers measuring the network's composition characteristic for the aforementioned AFs. The supposed superiority of composition characteristics of two different AFs used in tandem as opposed to using a single AF throughout is termed as '2 is better than 1' in the paper.

B. Contributions

To the best of our knowledge, there is no such study on using different AFs in hidden layers and switching them to achieve better performance. We propose and test the hypothesis '2 is better than 1' advocating the use of two different AFs in two hidden layers for smaller data and architecture and replicate the strategy in larger architectures by alternating AFs as the hidden layers increase. In the process, we contribute to the following:

- Prove Universal Approximation [11] mathematically (UAT) when AFs are switched i.e. in a compositional sense and hence show that a composition of AFs can approximate the non-linearity in deep neural networks.
- Conduct extensive empirical experiments on Computer Vision and Non-Computer vision data to validate our hypothesis on different optimizers.

The remainder of the paper is organized as follows. An intuition supported by a case study of two specific AFs used in the compositional sense is presented in Section III. This is followed by the Mathematical Setup in Section IV where we show the approximation ability of the network when two different AFs are used alternately. Section V contains detailed exposition on experiments and results on several benchmark data sets allowing us to conclude on the efficacy of the '2 is better than 1' hypothesis in section 6.

II. Intuition behind Our Hypothesis

Glorot and Bengio [12] hypothesized that in neural networks, the logistic layer output softmax(b+Wh) might initially rely more on the biases b, and hence push the activation value h towards 0, thus resulting in error gradients of smaller values. They referred to this as the saturation property of neural networks. This results in slower training and prevents the gradients from propagating backward until the layers close to the input learn. This saturation property is observed in the sigmoid. The sigmoid is non-symmetric around zero and hence obtains smaller error gradients when the sigmoid outputs a value close to 0. Similarly, tanh in all layers tends to saturate towards 1, which leads to layer saturation. All the layers attain a particular value, which is detrimental to the propagation of gradients. However, this issue of attaining saturation would be less pronounced in cases where two different activation functions are used. Since each activation function behaves differently in terms of gradients, i.e. sigmoid outputs are in the range [0,1], and the gradients are minimum at the maximum and minimum values of the function. Tanh on the other hand has minimum gradients at -1 and 1 and reaches its maximum at 0. Therefore, even if the layers begin to saturate to a common value, some of the layers would escape the saturation regime of their activations, and would still be able to learn essential features. As an outcome, this might result in fewer instances of vanishing gradients. This assumption would mean that networks with two different activations would learn faster and converge faster to a minima, and the same premise is supported with a Convergence study (details in section V). As demonstrated by Glorot and Bengio [12], if the saturation ratio of layers is less pronounced, it leads to better results in terms of accuracy. A standard neural network with N layers is given by,

$$h^l = \sigma(h^{l-1}W^l + b)$$

$$s^l = h^{l-1}W^l + b$$

where h^l is the output of the first hidden layer, σ is a non-linear activation function, and b is the bias. We compute the gradients as,

$$\frac{\partial Cost}{\partial s_k^l} = f'(s_k^l)W_{k,\cdot}^l \frac{\partial Cost}{\partial s^{l+1}}$$

$$\frac{\partial Cost}{\partial W_{m,n}^l} = z_l^i \frac{\partial Cost}{\partial s_k^l}$$

Now, we find the variances of these values. As the network propagates we need to ensure that the variances are equal to keep the information flowing. Essentially, when $\forall(l,l'), Var[h^l] = Var[h^{l'}]$, it ensures that forward propagation does not saturate, and when $\forall(l,l'), Var[\frac{\partial Cost}{\partial s^l}] = Var[\frac{\partial Cost}{\partial s^{l'}}]$, it ensures that backward propagation flows at a constant rate. Now, what remains is to calculate these variance values. The activation functions for which the derivative at 0 is 1, have been covered by Glorot and Bengio [12].

A Case Study

First, we attempt to find variance for two sigmoid activations in a network. The derivative of each activation output is approximately $0.25(\sigma'(0) = 0.25)$, as the weights are uniformly initialized, and the input features are assumed to have the same variance. Hence,

$$f'(s_k^l) = 0.25$$

$$Var[z^2] = Var[x]((0.25)^2 n_1 Var[W^{1'}] * (0.25)^2 n_2 Var[W^{2'}])$$

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

We see that this diminishing factor of 0.25^N steeply drops the variance during forward pass. Similarly, we observe that the gradient,

$$\frac{\partial Cost}{\partial s_k^l} = f'(s_k^l) W_{k,\cdot}^l \frac{\partial Cost}{\partial s^{l+1}}$$

has $f'(s_k^l)$ as one of the factors, and thus the diminishing factor is tied to the variance. Even when $N = 2$ the variance reduces by a factor of $4^4 = 256$.

Let's compute variance for a neural network with two hidden layers using sigmoid and tanh activations. For tanh, if the initial values are uniformly distributed around 0, the derivative is $f'(s_k^l) = 1$. Therefore, the variance for the second layer output is,

$$Var[z^2] = Var[x] * ((0.25)^2 * n_1 * Var[W^{1'}] * n_2 * Var[W^{2'}])$$

We see that the diminishing factor is just $4^2 = 16$, and this results in a much better variance when compared to the previous case. Therefore, using different AFs instead of the same implies a reduction in vanishing gradients and results in a much better flow of information because the variance value is preserved for longer.

III. Setup

Previous sections outlined the intuition behind using two different AFs alternately in the sense of function composition. The hypothesis '2 is better than 1' however rests on the theoretical guarantee that such compositions capture the latent non-linearity in data and provide good approximations via the composition of two different AFs. The different AFs in two layers form a composition function computed at individual neurons. If f(x) is applied at the first layer and g(x) at the second, then f(g(x) is considered a composition function. We use and extend the well-known results [13], [14] to establish the Universal Approximation Theorem (UAT) for the composition of AFs. We also exploit the existing results such as the following: An affine transformation of the plane has an inverse, that is also an affine transformation; and a composition of affine transformations is an affine transformation [13]. Theorem 1 proves that the composition of two AFs in a network with two hidden layers, with an arbitrary number of hidden units, is discriminatory and can approximate any function with a desired level of precision. Following similar lines, the values from the second hidden layer go to the third (the output layer), forming a composition function at the next level. Theorem 2 shows that a composition function at the output layer is discriminatory too. Following Guilhoto's argument [15], it is easy to follow that such an arrangement of alternating AFs would approximate the non-linearity over multiple hidden layers.

Universal Approximation Theorem for Sigmoidal Function

Definition 1. *A function $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoidal function if, for some finite constants $\mu_{+\infty}$ and $\mu_{-\infty}$ (which may take values as 0,1 or -1),*

$$\sigma(t) = \begin{cases} \mu_{+\infty} & t \to \infty \\ \mu_{-\infty} & t \to -\infty \end{cases}$$

Examples are logistic sigmoid, hyperbolic tangent and Rectified linear unit.

Definition 2. *Two hidden layer Feedforward Neural network (FNN) $\mathbb{N}$ and different sigmoidal Activation Function for σ sigmoid, T, R, input $x \in \mathbb{R}^d$, output $H(x) \in \mathbb{R}$, weight matrices, $W_1 \in \mathbb{R}^{n\mathbf{x}d}$ (weights connecting neurons of input and first hidden layer), $W_2 \in \mathbb{R}^{m\mathbf{x}n}$ (for first and second hidden layer), $W_3 \in \mathbb{R}^{1\mathbf{x}m}$ (for second hidden and output layer), $b_1 \in \mathbb{R}, b_2 \in \mathbb{R}$ the output of $\mathbb{N}$ is -*
$y = H(x) = \sigma(W_3\sigma'(W_2\sigma^*(W_1x+b_1)+b_2)+b_3)$ *where,* $f_1(x) = \sigma^*(W_1x+b_1)$ *and* $f_2(x) = W_3\sigma'(W_2\sigma^*(W_1x+b_1)+b_2)$.

Theorem 1. *We say any sigmoidal function $\sigma : \mathbb{R} \to \mathbb{R}$ is discriminatory if, for a measure $\mu \in M(I_n)$*

$$\int_{I_n} \sigma(W(f_2(x)+b))\, d\mu(x) = 0$$

$\forall W \in \mathbb{R}^{\mathbb{N}\mathbf{x}\mathbb{N}}, b \in \mathbb{R}$, implies $\mu = 0$. I_n denotes n-dimensional unit cube. $f_2(x)$ is a sigmoidal function of affine transformation of $f_1(x)$ which is also an affine transformation of input x.

Proof: Let $\sigma_N^*(x) = \sigma^*(N(W_1^T x + b_1)) \forall x, W_1, b_1 \in \mathbb{R}$ where σ_N^* is a sequence of sigmoidal functions s.t. $\sigma_N^* \to \sigma^*$. Additionally, σ_N^* converges point-wise and boundedly to σ^*. As $N \to \infty$, $\sigma_N^*(x) \to 1$ whenever $(W_1^T x + b_1 > 0)$. Similarly, $\sigma_N^*(x) \to 0$ when $(W_1^T x + b_1 < 0)$. And for the case $(W_1^T x + b_1 = 0)$, $\sigma_N^* = \sigma^*(b1)$. There exist some function $\delta(x)$, s.t.

$$\delta(x) = \begin{cases} 1 & W_1^T x + b_1 > 0 \\ 0 & W_1^T x + b_1 < 0 \\ \sigma(b_1) & W_1^T x + b_1 = 0 \end{cases}$$

By applying dominated convergence theorem, and defining M_0, M_1 as open half spaces for $(W_1^T x + b_1 < 0)$ and $(W_1^T x + b_1 > 0)$; M_2 to be a hyperplane that satisfies $(W_1^T x + b_1 = 0)$

$$\lim_{N\to\infty} \int_{I_n} \sigma_N^* d\mu(x) = \int_{I_n} \lim_{N\to\infty} \sigma_N^* d\mu(x) = \mu(M_1) + \sigma(b_1)\mu(M_2)$$

Now we consider a bounded linear functional $H : L^\infty(R) \to R$, and a measurable function f, such that $H(f) = \int_{I_n} f(W_1^T x) d\mu(x)$.

$$H(1_{[b_1,\infty)}) = \int_{I_n} 1_{[b_1,\infty)}\left(W_1^T x\right) d\mu(x) = \mu(M_1) + \mu(M_2) = 0$$

Likewise, for any open interval (b_1, ∞)

$$H(1_{(b_1,\infty)}) = \int_{I_n} 1_{(b_1,\infty)}\left(W_1^T x\right) d\mu(x) = \mu(M_0) = 0$$

This shows that for an indicator function on any interval and also for a simple function, H(f)=0. We know sin and cosine are elements of $L^\infty(R)$, so

$$\begin{aligned} H(cos) + iH(sin) &= \int_{I_n} \left(cos\left(W_1^T x\right) + isin\left(W_1^T x\right)\right) d\mu(x) \\ &= \int_{I_n} e^{iW_1^T x} d\mu(x) = 0 \end{aligned}$$

Since the Fourier transform of μ is zero, it means $\mu = 0$. Hence σ^* is discriminatory on a space of $C(I_n)$

in the output of two hidden layer given as $H(x) = \sigma(W_3\sigma'(W_2\sigma^*(W_1x+b_1)+b_2)+b_3)$. Furthermore, σ' is applied as an affine transformation of σ^*. Hence σ' is also discriminatory. Following the same rule, σ is discriminatory. ■

Theorem 2. *If f denotes any function in a space of continuous function $C(I_n)$, f is the supremum norm of the decision function, f. Let σ be any continuous, discriminatory function. Sum of functions resulting from the two hidden layers (M denotes number of neurons on second hidden layer) FNN is given by*

$$H(x) = \sum_{i=1}^{M} \lambda_i \sigma(W(f_2(x)+b))$$

then $H(x)$ is dense in $C(I_n)$, i.e. for $\psi > 0$, $\exists f$ s.t.

$$H(x) - f(x) \langle \psi$$

Proof: The proof can be seen from the Universal approximation theorem given by Cybenko [16] (Theorem 1). ■

IV. Experiments and Results

As the experiments are run in exhaustive and extensive manner, spanning several NCV and CV data sets, different combinations of AFs on different architectures (narrow and wide, deep neural nets, Convolutional NN and VGG16), there are several questions on the reporting structure of the results. In this section, we shall address the following questions:

- Is the proposed method consistent in outperforming (Loss and Accuracy) the traditional structure of deploying activations i.e. same AFs on all hidden layers?
- Over multiple runs on the same data, is the method stable in terms of accuracy and loss statistics (small variance from the mean)?
- Over multiple runs (10), is the method able to beat the other (standard) combinations of AFs consistently?
- Is the method able to converge faster in comparison to the threshold (best) accuracy achieved by other combinations of AFs?

As we shall observe, the above questions are satisfactorily addressed by the experiments, empirically confirming our hypothesis '2 is better than 1'. Note, for Computer Vision data, more than 2 hidden layers were used where 2 different AFs are alternately used in the hidden layers.

Settings and Evaluation

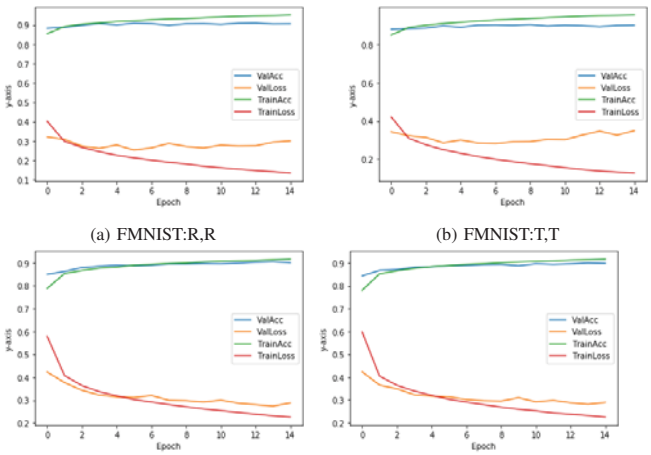
Our experiment involves 2-hidden layers NN (for NCV data) and deep NN architecture (Convolutional NN, VGG16), both implemented in Python3 by using Keras and Tensorflow built-in libraries. The experiments are performed on Google Colab notebook, run on (minimum) i5 processor and 8GB RAM. The model consisted of an input layer, 2-hidden layers and an output layer for NCV data. In the case of CV data, for MNIST, F-MNIST and Beans, 2 hidden Convolutional layers along with Max Pooling layers were used. As for CIFAR-10, the architecture consisted of an input layer, 8 CNN layers, and 2 Dense output layers, along with Batch Normalization and Max Pooling. For this network, the AFs were used in an alternating fashion. The VGG 16 architecture consists of 14 convolutional layers followed by 2 fully connected layers of size 4096 (where ReLU, Tanh and Sigmoid are plugged-in alternatively) which is later fed to the output layer with softmax activation function. The AFs used in the hidden layers are Sigmoid, Relu and Tanh. For each execution, different sets of AFs are plugged-in. The validation accuracy and loss are noted for each run. The NCV data sets used in the study are Iris, Wheat seeds, Bank Note Authentication, Diabetes, Mushroom, Heart Disease, Breast Cancer, Sonar, and zoo (Other CV datasets -Car Evaluation, Abalone, Haberman's Survival, Wine, Ionosphere, Titanic- used in the experiments). We also experimented on Computer Vision datasets like MNIST, FMNIST, CIFAR 10, and Beans. The model is trained with 5 fold cross-validation for every dataset to ensure it never overfits, followed by ten executions (25 and more epochs in each iteration). We calculated mean, standard deviation for accuracy, loss, and Bonferroni Criterion over 10 runs on each data set. Later we carried out a threshold-based study to understand the impact of two different AFs on convergence by comparing with a minimum threshold accuracy. We conducted experiments using fixed learning rates like $10^{-1}, 10^{-2}, 10^{-3}$ as well as learning rate decay [17] and found that the results were better and converged faster with the learning-rate-decay. Hence, the experiments in the manuscript are reported just for the Learning rate decay.

Performance Comparison: To interpret the results, we refer to Model A as one with two different AFs, and Model B with the same functions on two hidden layers. Table I gives the average results (training, validation accuracy, and loss) of 10 runs for each combination of AF, one dataset at a time. K-fold cross-validation is implemented to ensure that the model never overfits and the probability of bias gets eliminated. The tables refer to S, T and, R as Sigmoid, Tanh, and ReLU respectively, and S, T represents Sigmoid and ReLU on the 2 hidden layers. The same representations are used throughout the tables. On close examinations of the results in Table I, we infer that mostly all datasets returned greater accuracies for Model A (highlighted in bold). For example, Zoo, Heart disease shows 5-6% higher validation accuracies when Model A and B are compared for Sigmoid and Tanh. Similarly, 6-7% increase in accuracy is seen with Sigmoid and ReLU. For the remaining datasets, the accuracy of Model A is observed to be greater by 1-2% or at par with Model B. Interestingly, Mushroom and Bank Note data gave 100% accuracy for Model A (involving different AFs). Another interesting observation is the lowest loss (9.02E-06) reached with Model A, implemented with ReLU and Tanh on the mushroom dataset. Looking at the CV datasets, we saw that Sigmoid - Sigmoid networks fared worse than the other activation functions. However, in the case of Beans and Fashion-MNIST, we saw that Sigmoid-Tanh and Sigmoid-ReLU respectively had better accuracies and lower loss values, despite the poor performance of Beans on Sigmoid-Sigmoid.

Fig. 1 shows the accuracy comparison between the ReLU-Tanh and ReLU-ReLU activation functions in all the

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

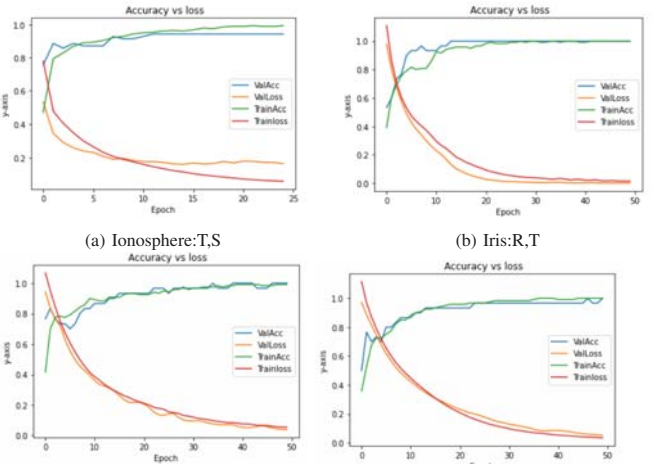
Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf



(a) FMNIST:R,R (b) FMNIST:T,T

(c) FMNIST:S,R (d) FMNIST:S,T

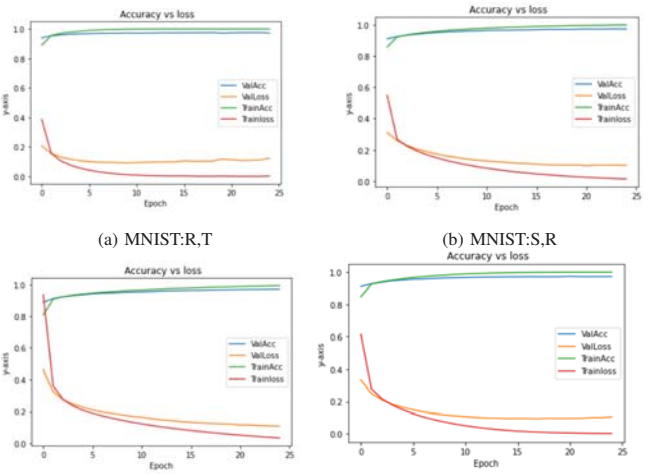
Fig. 1 Same AFs (Model B) in the models leading to a condition of overfit for R,R and T,T in FMNIST data; for Model A a fair training is noticeable; Plots of the Accuracy curve, loss plotted against the number of epochs; Caption of every sub-figure implies dataset: (AFs used)



(a) Ionosphere:T,S (b) Iris:R,T

(c) Iris:S,R (d) Iris:T,S

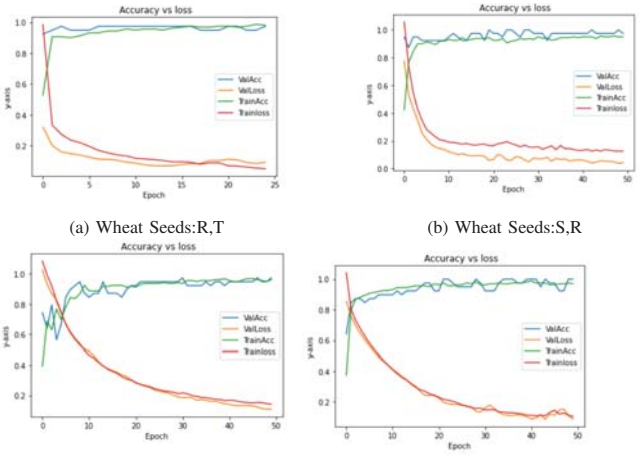
Fig. 2 Plots of the Accuracy curve, loss plotted against the number of epochs; caption of every sub-figure implies dataset:(AFs used)



(a) MNIST:R,T (b) MNIST:S,R

(c) MNIST:S,S (d) MNIST:T,S

Fig. 3 Plots of the Accuracy curve, loss plotted against the number of epochs; caption of every sub-figure implies dataset:(AFs used)



(a) Wheat Seeds:R,T (b) Wheat Seeds:S,R

(c) WheatSeeds:S,S (d) Wheat Seeds:T,S

Fig. 4 Plots of the Accuracy curve, loss plotted against the number of epochs; caption of every sub-figure implies dataset:(AFs used)

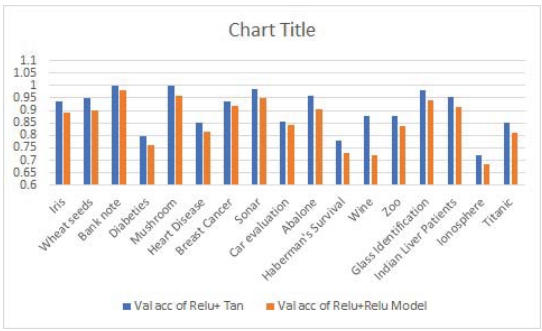


Fig. 5 Bar Plot to compare accuracy of the two Models; here we used values for (R,T) and (R,R) for all datasets; (R,T) worked as a better classifier than (R,R)

NCV datasets; which shows that ReLU-Tanh outperforms in all datasets. From the graphs in the figures, we see that most networks run on the same number of epochs are able to train without overfitting. Figs. 2 and 3 show that networks with the same activations start stagnating in their test loss while reducing train loss, whereas the networks with different activations are able to overcome this issue and train fairly. Fig. 4 shows the bar plots to compare the accuracies from the two models.

TABLE I
RESULTS OF NON-COMPUTER VISION AND COMPUTER VISION DATASETS, K-FOLD CROSS VALIDATED FOR 2 HIDDEN LAYERS NN SHOWING TRAIN AND VALIDATION ACCURACIES AVERAGED OVER 10 EXECUTIONS

Non Computer Vision Datasets (Optimizer - Adam)								
	Model A - Two different AFs				Model B - Same AFs			
Datasets	AFs (Model A)	Avg.Tr.$Acc \pm SD$	Avg.Val.$Acc \pm SD$	Loss	AFs (Model B)	Avg.Tr.$Acc \pm SD$	Avg.Val.$Acc \pm SD$	Loss
Zoo	S,T	1 ± 0	**0.93 ± 0.018**	0.25	R,R	0.88 ± 0.001	0.87 ± 0.001	0.28
	R,S	1 ± 0	**0.94 ± 0.012**	0.062	S,S	0.86 ± 0.001	0.86 ± 0.001	0.29
	R,T	1 ± 0	**0.94 ± 0.012**	0.20	T,T	0.88 ± 0.001	0.87 ± 0.014	0.27
Heart Disease	S,T	0.94 ± 0.017	**0.84 ± 0.03**	0.45	R,R	0.88 ± 0.004	0.81 ± 0.002	0.44
	S,R	0.86 ± 0.001	**0.82 ± 0.001**	0.41	S,S	0.86 ± 0.002	0.81± 0.001	0.42
	T,R	0.93 ± 0.004	**0.87 ± 0.03**	0.46	T,T	0.89 ± 0.001	0.79 ± 0.001	0.55
Sonar	S,T	0.97 ± 0.03	**0.81 ± 0.001**	0.003	R,R	1 ± 0.001	0.84 ± 0.02	0.001
	R,S	1 ± 0	**0.84 ± 0.06**	0.007	S,S	0.96 ± 0.001	0.78 ± 0.01	0.001
	R,T	1 ± 0	**0.84 ± 0.01**	0.002	T,T	1 ± 0	0.80 ± 0.02	0.004
Iris	S,T	0.88 ± 0.01	0.86 ± 0.01	0.41	R,R	0.93 ± 0.009	0.88 ± 0.01	0.09
	R,S	0.97 ± 0.03	**0.95 ± 0.01**	0.20	S,S	0.76 ± 0.01	0.74 ± 0.02	0.77
	R,T	0.99 ± 0.00	**0.97 ± 0.01**	0.15	T,T	0.94 ± 0.01	0.91 ± 0.01	0.26
Wheat Seeds	S,T	0.96± 0.01	**0.94 ± 0.01**	0.26	R,R	0.97 ± 0.001	0.90 ± 0.01	0.21
	R,S	0.96 ± 0.001	**0.94 ± 0.001**	0.20	S,S	0.94 ± 0.001	0.93 ± 0.01	0.24
	R,T	0.98 ± 0.001	**0.95 ± 0.001**	0.19	T,T	0.97 ± 0.001	0.94 ± 0.001	0.16
Bank Note	S,T	1 ± 0	**1 ± 0**	0.0	R,R	1 ± 0	1 ± 0	0
	R,S	1 ± 0	**1 ± 0**	0.0	S,S	1 ± 0	0.99 ± 0.001	0
	R,T	1 ± 0	**1 ± 0**	0.0	T,T	1 ± 0	0.99 ± 0.001	0
Diabetes	S,T	0.78 ± 0.001	**0.77 ± 0.001**	0.47	R,R	0.80 ± 0.001	0.75 ± 0.001	0.49
	S,R	0.78 ± 0.002	**0.77 ± 0.007**	0.46	S,S	0.78 ± 0.003	0.76 ± 0.001	0.48
	R,T	0.80 ± 0.008	**0.76 ± 0.009**	0.47	T,T	0.79 ± 0.005	0.76 ± 0.004	0.48
Mushroom	S,T	1 ± 0	**1 ± 0**	2.13E-05	R,R	1 ± 0	1 ± 0	1.63E-05
	R,S	1 ± 0	**1 ± 0**	2.55E-05	S,S	0.99 ± 0.001	0.99 ± 0.001	2.69E-03
	R,T	1 ± 0	**1 ± 0**	9.02E-06	T,T	1 ± 0	1 ± 0	1.09E-05
Breast Cancer	S,T	0.97 ± 0.005	**0.96 ± 0.004**	0.23	R,R	0.97 ± 0.007	0.96 ± 0.001	0.24
	S,R	0.97 ± 0.004	**0.97 ± 0.007**	0.19	S,S	0.82 ± 0.001	0.82 ± 0.01	0.2
	R,T	0.97 ± 0.003	**0.97 ± 0.002**	0.25	T,T	0.96 ± 0.001	0.96 ± 0.001	0.16
Car Evaluation	S,T	0.93 ± 0.008	0.91 ± 0.009	0.20	R,R	0.92 ± 0.004	0.90 ± 0.005	0.02
	R,S	0.96 ± 0.001	**0.95 ± 0.001**	0.11	S,S	0.83 ± 0.003	0.82 ± 0.005	0.41
	R,T	0.93 ± 0.01	0.91 ± 0.01	0.20	T,T	0.96 ± 0.009	0.94 ± 0.01	0.15
Abalone	S,T	0.78 ± 0.001	**0.77 ± 0.002**	0.47	R,R	0.76 ± 0.09	0.77 ± 0.002	0.47
	S,R	0.79 ± 0.002	**0.78 ± 0.002**	0.47	S,S	0.78 ± 0.001	0.77 ± 0.003	0.46
	R,T	0.80 ± 0.003	**0.78 ± 0.002**	0.50	T,T	0.78 ± 0.002	0.77 ± 0.002	0.47
*HS**	S,T	0.76 ± 0.001	**0.74 ± 0.002**	0.55	R,R	0.79 ± 0.001	0.73 ± 0.001	0.55
	S,R	0.76 ± 0.001	**0.75 ± 0.011**	0.54	S,S	0.75 ± 0.001	0.74 ± 0.001	0.54
	T,R	0.78 ± 0.001	**0.74 ± 0.002**	0.52	T,T	0.77 ± 0.001	0.73 ± 0.001	0.55
Wine	S,T	0.89 ± 0.004	**0.88 ± 0.002**	0.27	R,R	0.89 ± 0.001	0.88 ± 0.003	0.28
	S,R	0.88 ± 0.001	**0.88 ± 0.001**	0.27	S,S	0.89 ± 0.001	0.87 ± 0.006	0.29
	T,R	0.89 ± 0.011	**0.88 ± 0.015**	0.27	T,T	0.89 ± 0.001	0.88 ± 0.001	0.27
Ionosphere	S,T	0.92 ± 0.005	0.88 ± 0.010	0.29	R,R	0.98 ± 0.002	0.91 ± 0.01	0.23
	R,S	0.97 ± 0.006	**0.91 ± 0.009**	0.22	S,S	0.90 ± 0.002	0.86 ± 0.008	0.32
	R,T	0.98 ± 0.008	**0.92 ± 0.001**	0.23	T,T	0.97 ± 0.002	0.91 ± 0.007	0.23
Titanic	S,T	0.80 ± 0.002	**0.79 ± 0.003**	0.46	R,R	0.83 ± 0.004	0.81 ± 0.004	0.44
	R,S	0.82 ± 0.003	**0.81 ± 0.005**	0.44	S,S	0.80 ± 0.002	0.79 ± 0.005	0.45
	R,T	0.83 ± 0.002	**0.81 ± 0.006**	0.43	T,T	0.82 ± 0.002	0.79 ± 0.004	0.42
Computer Vision Datasets (Optimizer - Adam)								
	Model A - Two different AFs				Model B - Same AFs			
Datasets	AFs (Model A)	Avg.Tr.$Acc \pm SD$	Avg.Val.$Acc \pm SD$	Loss	AFs (Model B)	Avg.Tr.$Acc \pm SD$	Avg.Val.$Acc \pm SD$	Loss
MNIST	S,T	0.995 ± 0.0005	**0.972 ± 0.0004**	0.10	R,R	0.997 ± 0.0001	0.9748 ± 0.0006	0.15
	R,S	0.9997 ± 0.0003	**0.976 ± 0.0004**	0.18	S,S	0.9995 ± 0.00	0.9701 ± 0.0008	0.13
	R,T	0.9997 ± 0.0004	**0.974 ± 0.001**	0.11	T,T	0.9948 ± 0.001	0.969 ± 0.001	0.10
CIFAR-10**	R,T	0.95 ± 0.001	**0.87 ± 0.002**	0.55	R,R	0.96 ± 0.001	0.86 ± 0.001	0.62
					T,T	0.60 ± 0.001	0.59 ± 0.001	1.17
F-MNIST	S,T	0.93 ± 0.0011	**0.90 ± 0.052**	0.27	R,R	0.95 ± 0.001	0.90 ± 0.001	0.29
	R,S	0.94 ± 0.001	**0.90 ± 0.019**	0.29	S,S	0.89 ± 0.002	0.88 ± 0.002	0.32
	R,T	0.96 ± 0.001	**0.90 ± 0.001**	0.32	T,T	0.96 ± 0.001	0.90 ± 0.001	0.34
Beans	S,T	0.96 ± 0.02	0.69 ± 0.02	0.91	R,R	0.89 ± 0.004	0.73 ± 0.001	0.91
	R,S	0.96 ± 0.01	**0.75 ± 0.01**	1.01	S,S	0.34 ± 0.02	0.31± 0.01	1.1
	R,T	0.95 ± 0.02	**0.74 ± 0.01**	0.93	T,T	0.98 ± 0.02	0.972 ± 0.01	1.23
MNIST (VGG16)	S,T	0.91 ± 0.01	**0.94 ± 0.02**	0.45	R,R	0.92 ± 0.005	0.91 ± 0.03	0.27
	R,S	0.97 ± 0.03	**0.95 ± 0.01**	0.20	S,S	0.82 ± 0.02	0.89 ± 0.03	0.40
	R,T	0.91 ± 0.04	**0.90 ± 0.02**	0.27	T,T	0.90 ± 0.01	0.90± 0.02	0.29

Loss is also reported. S- sigmoid; T-tanh; R-ReLU;CIFAR-10** - CIFAR could not be trained and gave very low accuracy for R,S;S,T; S,S.

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

Robustness Study: Comparing the efficacy of Model A with B over a single run is not significant statistically. Hence we incorporated Bonferroni Criterion (BC) to the effectiveness of Model A over ten successive runs. BC gives a statistical count of the number of times Model A is better than B when compared over ten runs (Table II). To ensure a fair comparison of the two models, we computed BC1 and BC2. The interpretation of BC1 and BC2 is as follows. We know that Model A has different AFs while Model B has the same AF on hidden layers. BC1 denotes a count of the number of times the accuracy of Model A surpassed B when Model A's first AF is deployed in B. For instance, when comparing Sigmoid and ReLU (S, R), BC1 signifies the number of times (S,R) beats the accuracy of (S,S) in 10 executions. Furthermore, in Table II, a BC1 value of 10 (say, Mushroom dataset) suggests that (S,R) has surpassed (S,S) in all the 10 executions. Likewise, BC2 signifies the number of times the accuracy of (S,R) surpassed (R,R). For example, a BC2 value of 9 (Iris data) infers (S,R) exceeds (R,R) in accuracy, 9 times out of 10. These comparisons are carried out for each possible combination of AF's (S,R), (R,T), and (T,S) on every dataset. We observed from Table II, Model A performed better than B in Mushroom, Titanic, Zoo, Banknote, Wine, Diabetes, Iris, and Heart Disease. As a rule of thumb, a BC value larger than 6 conveys Model A performing better than B, and Table II shows most of the BC values larger than 6. As far as the CV datasets are concerned, (S,R) or (R,T) surpassed the competitor configurations very promptly, particularly in all cases of (S,S) and (T,T). The table shows promising results especially for Beans and MNIST, where the (S,R) and (S,T) networks performed much better than their counterparts. In CIFAR-10, (T,R) is marginally better than (R,R).

Convergence Study

It is important to compare the convergence of both models while looking into their accuracy and losses. Is there a substantial difference in the convergence (no. of epochs) when both AFs are deployed in Model A and when just one AF was used? We went ahead in our experiments to perform a thresholding study of epochs needed to reach the maximum accuracy for both models. We picked two AFs to begin with (say S,T) and made combinations under Model A (S,T) and B (S,S, T,T). We ran NN for 25-50 epochs with activations from the above combinations and recorded 'threshold accuracy (ThAC)' defined as the minimum accuracy reached by the above combinations. We also recorded the epochs needed to reach ThAC for every NN under Model A and B. The results were very encouraging and interesting to explore. The threshold study performed for Iris data revealed 74% accuracy ($ThAC_{Iris} = 74\%$) reached for (S,S) at 49 epochs. Interestingly, epochs that needed to reach 74% by (S,T) model were just 22. This indicates that Model A reaches the accuracy set by Model B in less than half of the epochs. Besides, (T,T) architecture converged at 47 epochs with ThAC of 91% while (R,T) model converged at just 16 epochs (one third) while reaching 91% accuracy. Additionally, (S,R) took 7 epochs to reach ThAC, whereas Model B (S,S) took 24 epochs to converge at ThAC. Similar trends were witnessed on the remaining datasets. For Fashion MNIST, (S,S) took 15 epochs to converge at accuracies around 89%, whereas (R,S) and (T,S) took 9 epochs to reach ThAC. In Beans, (S,S) had poor training outcomes even after 15-20 epochs, whereas (R,S) and (T,S) managed to cross ThAC within the first 2 epochs. *We observe a consistency in faster convergence when 2 different AFs are used, across 19 data sets.* A complete tabulation of threshold convergence is available at github repository which can be shared on request. Additionally, our model on average yielded 9% greater accuracy on the Iris dataset (with ADAM optimizer) in comparison with several other SoTA optimizers such as AdaGrad, AdaDelta, AMSgrad, and AdaSwarm [18]. Similarly, the accuracy on Pima India Diabetes dataset is 6-13% greater; for Wheat Seeds, the accuracy obtained is 13-28% higher; for Heart Disease the accuracies varied to a greater range of 11-38% when compared with AdaSwarm and AdaGrad optimizers. There are also datasets like Car Evaluation with accuracies varied over a range of 4-10% and it was observed that our model produces the best results when compared to other optimizers. The corresponding results are presented in Table I.

TABLE II
BONFERRONI CRITERION (BC) FOR ALL COMBINATIONS OF AFS
[T-TANH S-SIGMOID, R-RELU]

Model A- Two different AFs	S,R		S,T		T,R	
Model B - Same AFs	S,S	R,R	S,S	T, T	T, T	R,R
Datasets	BC1	BC2	BC1	BC2	BC1	BC2
Mushroom	**10**	**10**	**10**	**10**	**10**	**10**
Titanic	**10**	**10**	**10**	**10**	**10**	6
Zoo	**10**	8	**10**	8	**10**	**10**
Heart Disease	**10**	8	9	9	**10**	**10**
Bank Note	8	**10**	**10**	**10**	**10**	**10**
Wine	**10**	7	**10**	**10**	**10**	7
Diabetes	7	9	**10**	**10**	10	10
Iris	**10**	9	**10**	10	6	10
Wheat Seeds	**10**	9	6	7	8	**10**
Breast Cancer	**10**	8	8	6	**10**	8
Sonar	**10**	**10**	10	7	10	7
Car Evaluation	**10**	**10**	**10**	4	4	8
Abalone	5	5	8	8	8	9
Haberman's Survival	8	8	5	9	9	9
Ionosphere	**10**	3	8	6	**10**	3
Cifar-10	-	10	-	10	**10**	6
Fashion-MNIST	**10**	0	**10**	4	**10**	3
Beans	**10**	9	**10**	2	**10**	6
MNIST	**10**	2	**10**	**10**	**10**	3

V. CONCLUSION

The manuscript broadly examines the effectiveness of a composition of two different activation functions on the expressive power of NN by analyzing it on a large number of NCV and CV datasets. Our hypothesis '2 is better is 1' is undoubtedly supported by all the datasets used in the study. The claim is that the simple, non-linear AFs together in a network may create complex but more effective composition functions, thus leading to better classification metrics. A 2-hidden-layer NN and deep learning network (Convolutional NN, VGG16) were run for the standard benchmark datasets from the UCI Machine Learning repository to test our hypothesis and, the Models with different AFs surpassed

Open Science Index, Computer and Systems Engineering Vol:17, No:11, 2023 publications.waset.org/10013347.pdf

the ones that used the same with a margin of 6-7%. We investigated the deep learning architectures to explore the hypothesis and found that accuracies for Model A were 1-3% larger than B for the benchmark CV datasets. In total, we experimented on 19 different data of all types and reported mean and standard deviation of classification metrics resulting from the ten runs from every composition [(S,T), (T,R), (R,S), so on]. The risk of overfitting and bias is kept at bay by assuring K-fold cross-validation in each run, the same is validated in the training-validation plots. To thoroughly test the hypothesis, we compared models with all possible options, say, if (R,T) is studied on some data, its metrics are compared against (R,R) and (T,T). We found that metrics for (R,T) and (T,R) were close, assuring empirical commutativity. The robustness of the hypothesis was assessed by Bonferroni criteria, and we inspected the two BC's to be larger for Model A. In particular, Mushroom, Banknote, and many datasets yielded BC values of 10. It was also found that the 2 different AF combinations converged to threshold accuracy faster.

The empirical study is based on the fact that two different AFs generalize better over two or more hidden layer architectures. This was argued effectively and theoretical results on the approximation ability of compositional activations when used alternately have been established. We restricted the current study to sigmoid, ReLU, tanh and their combinations and evaluated the hypothesis for these simple yet popular AFs. However, we would like to extend this work for other AF combinations that include SELU, ELU, Swish, Mish and many more. The study is encouraging enough to pursue in future and investigate compositional activations for more complex data and architectures.

Acknowledgments

Archana Mathur would like to thank SERB for funding under the "Teachers Associate-ship for Research Excellence" (SERB File Number: TAR/2021/000206) scheme to carry out this research. Snehanshu Saha would like to thank the Science and Engineering Research Board (SERB)-DST (File SERB-EMR/ 2016/005687), Government of of India for supporting this research at BITS Pilani K.K Birla Goa Campus.

References

[1] M. Mohammadzaheri, L. Chen, A. Ghaffari, and J. Willison, "A combination of linear and nonlinear activation functions in neural networks for modeling a de-superheater," *Simulation Modelling Practice and Theory*, vol. 17, no. 2, pp. 398–407, 2009. *Available at https://www.sciencedirect.com/science/article/abs/pii/S1569190X08001913*.

[2] J. Feng and S. Lu, "Performance analysis of various activation functions in artificial neural networks," in *Journal of physics: conference series*, vol. 1237, p. 022030, IOP Publishing, 2019. *Available at https://iopscience.iop.org/article/10.1088/1742-6596/1237/2/022030*.

[3] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *Journal of theoretical and applied information technology*, vol. 47, no. 3, pp. 1264–1268, 2013. *Available at http://www.jatit.org/volumes/Vol47No3/61Vol47No3.pdf*.

[4] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018. *Available at https://arxiv.org/abs/1811.03378*.

[5] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011. *Available at https://www.researchgate.net/publication/228813985_Performance_Analysis_of_Various_Activation_Functions_in_Generalized_MLP_Architectures_of_Neural_Networks*.

[6] S. Saha, A. Mathur, A. Pandey, and H. Arun Kumar, "Diffact: A unifying framework for activation functions," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989. *Available at https://www.sciencedirect.com/science/article/abs/pii/0893608089900208*.

[8] J. Makhoul, R. Schwartz, and A. El-Jaroudi, "Classification capabilities of two-layer neural nets," in *International Conference on Acoustics, Speech, and Signal Processing,*, pp. 635–638, IEEE, 1989. *Available at https://www.semanticscholar.org/paper/Classification-capabilities-of-two-layer-neural-Makhoul-Schwartz/c42c1305ce33c628ffc5401d5de2b0347f50ac78*.

[9] G. R. Brightwell, C. Mathieu, and H. Paugam-Moisy, "Multilayer neural networks: One or two hidden layers?," in *NIPS*, 1996.

[10] W. Wan, S. Mabu, K. Shimada, K. Hirasawa, and J. Hu, "Enhancing the generalization ability of neural networks through controlling the hidden layers," *Appl. Soft Comput.*, vol. 9, pp. 404–414, 2009.

[11] S. Saha, N. Nagaraj, A. Mathur, R. Yedida, and S. H. R, "Evolution of novel activation functions in neural network training for astronomy data: habitability classification of exoplanets," *The European Physical Journal. Special Topics*, vol. 229, pp. 2629 – 2738, 2020.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[13] R. Yedida and S. Saha, "Beginning with machine learning: a comprehensive primer," *The European Physical Journal Special Topics*, 2021.

[14] S. Saha, A. Mathur, K. Bora, S. Agrawal, and S. Basak, "A new activation function for artificial neural net based habitability classification," *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1781–1786, 2018.

[15] L. F. Guilhoto, "An overview of artificial neural networks for mathematicians," 2018.

[16] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989. *Available at https://link.springer.com/article/10.1007/BF02551274*.

[17] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," 2019.

[18] S. S. Dhavala and S. Saha, "Adaswarm: Augmenting gradient-based optimizers in deep learning with swarm intelligence," *Available at https://ieeexplore.ieee.org/abstract/document/9472873*.