PERFORMANCE OF NOVEL ACTIVATION FUNCTIONS ON DEEP LEARNING ARCHITECTURE

Snehanshu Saha
BITS Pilani, K K Birla Goa
255 PUBLICATIONS **1,369** CITATIONS

Analysing the Performance of Novel Activation Functions on Deep Learning Architectures

Animesh Chaturvedi, Apoorva N, Mayank Sharan Awasthi, Shubhra Jyoti, Akarsha D P, Soumya, and Brunda S

Nitte Meenakshi Institute of Technology

Abstract. Deep learning is a cutting-edge technology that functions similarly to the human nervous system. Neural networks are at the heart of Deep Learning. Neural networks are made up of numerous layers, including the input layer, which accepts raw data as input, hidden layers, which process the input data, and a final layer, the output layer, which provides the result. Its workflow pattern is comparable to machine learning [2], [11], allowing us to gain hands-on expertise with this technology, speed up our work, and allow us to make several efforts without having to develop a basic Machine learning algorithm from scratch. In the case of deep learning, there are several neural networks to choose from. The majority of Deep Learning architectures are built on neural networks such as CNN, RNN, and others. Deep neural network activation function development is often guided by set goals and gradual steps toward tackling specific challenges. The primary goal of this study is to examine the performance of innovative activation functions (SBAF parabola [6] [16], AReLU [7], Leaky ReLU, SWISH) on deep learning architectures such as CNN, DENSENET, etc. On deep learning architectures, our study will compare the classification performance of the aforementioned activation functions. Our code:
-allows the user to run DL architectures on selected computer vision datasets using selected activation functions;
-allows the user to display classification accuracies and other performance parameters;
-reports different validation curves to validate system performance; -runs the code for multiple loss functions

Keywords: Embedded systems · Redundancy · Robotics · Network reliability.

1 Motivation

Deep learning is becoming increasingly popular in a variety of fields, including medicine, the military, and aerospace. Deep Learning delivers the highest accuracy when trained on large amounts of data, which contributes to its widespread appeal. It has a significant benefit, and this plays a major role in explaining its widespread popularity. Now, the "Big Data Era" [13] presents a plethora of options to develop new technologies and promote its appeal. From efficient inference on small data [14], Deep learning evolved in a manner comparable to the human nervous system, operating on huge data sets, solving a range of problems from estimation and forecasting [15] to classification that includes Self-driving vehicles, translation, and revolutionary medical treatments. These are testimony of deep learning capabilities and its broad range of exploration possibilities. The unique activation functions are being utilised and investigated on various data sets for exoplanet classification [18], [19] . When compared to the standard ReLU and Sigmoid Activation Functions, these activation functions SBAF parabola, AReLU, SWISH, and LReLU performed incredibly well on Vanilla Neural Networks and provided close to 99% accuracy on various datasets. It will be fascinating to observe if these activation functions perform similarly well for Deep Learning architectures such as CNN [9], DENSENeT, Imagenet, and so on.

2 Activation functions

Activation functions serve as a link between the data sent to the input layer and the neuron presently in use, as well as the outcomes sent to the final output layer. Neuron activation is determined by computing the weighted sum of activation functions and then adding bias to the total. Neurons with similar information are triggered. The activation of neurons is governed by a set of principles. The primary goal of activation functions is to introduce non-linearity. The outcomes of activation functions are passed to the next layer in forward propagation. If the produced value differs significantly from the real value, an error is calculated. The process is then referred to as reverse propagation. Consider the neural network shown below: Elements of network:

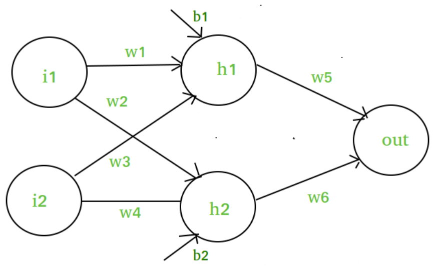


Fig. 1: Example of Neural Network

The above network layers can be pictured as-
Layer1:
Y(1) = W(1)M + b(1)
c(1) = Y(1)
where, Y(1) stands as result of layer 1

W(1) is that the weights allotted to neurons of hidden layer i.e. w1, w2, w3 and w4 and M be the input options i.e. i1 and i2
b is the vectored bias assigned to neurons in hidden layer i.e. b1 and b2
c(1) is that the vectored sort of any linear function.

Layer2:
Y(2) = W(2)c(1) + b(2)
c(2) = Y(2)
// substitute Y(1) here
Y(2) = (W(2) * [W(1)M + b(1)]) + b(2)
Y(2) = [W(2) * W(1)] * M + [W(2)*b(1) + b(2)]
Let,

[W(2) * W(1)] = w [W(2)*b(1) + b(2)] = b
Result: Y(2) = W*M + b ,result obtained is a linear function.

If activation functions do not appear to be applied, then the equation of weights and bias must be linear. Solving linear equations is straightforward, however it has a limited capability in determining complex difficulties that require complex understanding. Any neural network that lacks an activation function represents a linear regression model.

It is advantageous to employ non-linear activation functions, which aid in the analysis of complicated data, computation, and learning, as well as the generation of proper outputs.Despite the fact that there are many layers in the network, using a linear activation function yields another linear equation since concatenating them yields another linear equation, which is insufficient to enhance the model. Because the derivative of the linear activation function yields a constant, it is not feasible to go back and checkout neurons.
There are a number of activation functions -

2.1 Rectified Linear Unit (ReLU)

ReLU seems to be a linear activation function, yet it allows the network to converge fast. However, because there is a derivative, backpropagation is attainable. Because a model that utilises it is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks.

$$f(x) = x^+ = \max(0, x) \text{ where x is the input to a neuron.}$$

This is also known as a ramp function.

2.2 Attention-based-Rectified-Linear-Unit (AReLU)

When the attention module is combined with a rectified linear unit (ReLU), positive aspects are amplified and negative elements are suppressed, both with learnt, data-adaptive parameters.AReLU performs well when learning rates are low, making it ideal for transfer learning and making network training more resistant to gradient vanishing. AReLU is written as:

$$y = kx^n$$

2.3 Saha-Bora Activation Function (SBAF) [1]

The effectiveness of a new activation function in Artificial Neural Networks (ANN) in classifying exoplanets is investigated.The impetus for the Saha-Bora Activation Function (SBAF) stems from a long-standing awareness of the use of advanced mathematics in calculating exoplanet habitability scores. The function is shown to have good analytical characteristics and does not appear to have any local oscillation issues.Saha-Bora Activation Function is formulated as :

$$y = \frac{1}{1+kx^{\alpha}(1-x)^(1-\alpha)}$$

2.4 Swish [17]

The first thing that comes to mind while looking at this layout is that it looks quite a bit like ReLU, with one exception: the domain about 0 is not the same as ReLU.It is a non-obtrusive function. That is, unlike ReLU, it does not abruptly shift direction near x = 0. However, it bends smoothly from 0 and higher and then back upwards. It is also non-monotonic as a result of this fact. As a result, unlike ReLU and the other two activation functions, it does not remain steady or move in one direction. According to the authors, it is this trait that distinguishes Swish from most other activation functions, which share this uniformity.Swish Activation Function is uninterrupted at all points. It is termed as-

$y = swish(x) = x\sigma(\beta x)$ where $\sigma(x) = 1/(1 + exp(-x))$, is the sigmoid function.

βcan either be a constant defined prior to the training or a parameter that can be trained during training time. The derivative of the Swish Activation function is-

$$f'(x) = \beta f(\beta x) + \sigma(\beta x)(1–\beta f(\beta x))$$

2.5 Scaled Exponential Linear Unit (SeLU)

SeLU causes neural networks to have a self-normalizing feature. The activations of neurons converge towards a zero mean and unit variance. Because SeLUs have such a high level of self-normalization, we no longer have to be concerned with disappearing gradients.It is formulated as

$if a > 0 : returnscale * a$
$if a < 0 : returnscale * alpha * (exp(a) - 1)$
alpha and scale are constants
(alpha=1.67326324 and scale=1.05070098).

The values for alpha and scale are chosen in such a manner that the mean and variance of the values supplied as input are maintained throughout all layers and the inputs are appropriately big.

2.6 Exponential Linear Unit(ELU)

ELU, which stands for Exponential Linear Unit, is an activation function with comparable functionality to ReLU but less differences. Unlike other activation functions, ELU does not have the issue of disappearing and bursting gradients. It is uninterruptible and distinguishable at all places.

$$y = ELU(x) = exp(x)\neg 1; if x < 0$$
$$y = ELU(x) = x; if \geq 0$$

2.7 Mish

Mish is a self-regularized non-monotonic activation function inspired by Swish's self-gating feature. In LSTMs and Highway Networks, self-gating is the use of the sigmoid function. It is an uninterruptible, smooth, non-monotonic activation function. It can be formulated as :

$$f(x) = xtanh(softplus(x)) = xtanh(ln(1 + e^x));$$

$$f'(x) = \frac{e^x \omega}{\delta^2}$$
where, $\omega = 4(x + 1) + 4e^2x + e^3x + e^x(4x + 6)$and $\delta = 2e^x + e^2x + 2$

3 Different types of neural network

The design of neural network is similar to human nervous system.It comprises of one input layer, two or more hidden layers,and finally one output layer.There are various types of neural networks like feed-forward, CNN, Rnn, etc. Different types of network have their own business purpose.

3.1 CONVOLUTION NEURAL NETWORKS

First let us discuss about stride and cushioning first. A filter is a set of weights in the form of a matrix that is applied to an image or a matrix to create the desired features; it can be of any depth 'd'.
Striding: We don't always need all of the cells, and we sometimes leave out a few.
Padding occurs when the resulting output dimensions do not match the input dimensions during convoluting. Some data will be lost across borders, therefore we add zeros and recalculate them. The input in this case is an image with dimensions of height=28, width=28, and depth=1.

1st Layer Convolution:
In this step, input is taken and convoluted with 5 x 5 filters. It produces an output of size 28 x 28, and 6 filters are employed. As a result, the depth of convolution1 is 6. As a result, 24 x 24 x 6 gets transmitted to the next layer.
Pooling layer (the second layer):
In this level, the pooling layer receives an input of 24 x 24 x 6. Average pooling is accomplished here using a matrix of 22 and a stride of 2, i.e. setting a 2 x 2 matrix on an input and taking the average of all four pixels and jumping with a 2 column skip each time. As a result, the outcome is 12 x 12 x 6. Every layer has been calculated, and the depth is 6.
Layer 2 of convolution (third layer): In this level, a 12 x 12 x 6 matrix is used as an input, i.e. the results of the previous layer are taken and convoluted with a filter of size 5 x5, a stride of 1 (there is no skip), and zero padding. As a result, we receive a 5 x 5 output. 16 such filters of depth 6 is taken and after convoluting generates a result of 5x 5 x 16.
Layer 2 (fourth layer) pooling:
In this stage, the output of the preceding layer is taken and average pooling is conducted with a stride of 2 (skip two columns) and a filter of size 2 x 2, which is placed on the 5 x5 x 16 layers, resulting in 4x4 outputs for each 5x5. As a result, we obtain 4x4x16. Finally, we flatten them by feeding them into a feed-forward network.

3.2 DENSENET

Densely Connected Convolutional Networks i.e DenseNets, are the new choice of the day. When we go further, we find that CNN has flaws. Because the route for information to go from the initial input layers to the final output layer (and for the gradient in the other direction) is longer and larger, they disappear before reaching the other side of the layer. Densenet is similar to resnet but achieves more precision. In other networks, the results of the preceding layer are linked to the next layer via a sequence of processes. Convolution operations on pooling layers, batch normalisation, and an activation function are often included in the collection of procedures.
Densenets, on the other hand, do not combine the layer's feature maps with the input features of other layers, but rather concatenate them.

The equation reshapes again into: $x_l = H_l(x_0, x_1, x_2, ..., x_l - 1)$

DenseNets are seen as groups of DenseBlocks, with the number of filters changing but the size of the maps remaining constant. Transition layers are the layers that exist between these blocks. Normalizing layers by re-centering and re-scaling inputs stabilises neural networks. The preceding layers' results are normalised by executing 1x1 convolution and 2x2 pooling layers. We concatenate the outputs of all layers here, thus dimensions are increased at all levels. The generalisation at the kth layer following H1 that produces'm' features every time is known as:

$m_l = m_0 + m * (k-1)$,here m is growth rate.

The following layers add additional functionality to the layers that came before them. Transition blocks execute 1*1 convolution with filters, followed by 2*2 pooling with a stride of 2, resulting in a 50 percent decrease in feature maps and size.As we progress through the network, the number of levels increases 'm' times. Initially, 1*1 convolution is conducted using 128 filters, followed by 3*3 convolution. This is accomplished through the usage of 32 feature maps. This is known as the growth rate.
Finally, each layer in each denseblock performs the identical set of operations on the input, and the resulting outputs are concatenated. Each layer contributes fresh information to the ones that came before it.

4 Dataset

- Cifar 100 : CIFAR 100 is identical to CIFAR 10, except it has 100 classes with 600 photos in each class. There are 500 training photos and 100 testing images in each class. CIFAR 100 consists of 100 classes divided into 20 superclasses. Each picture has a "fine" label and a "coarse" label. Fine labels define the class to which they belong, whereas coarse labels identify the superclass to which they belong.
- MNIST: MNIST stands for Modified National Institute of Standards and Technology. This collection includes 60000 tiny square 28*28 pixel grayscale pictures of handwritten single numerals ranging from 0 to 9.
- Fashion MNIST: Fashion MNIST is a dataset that contains 60000 examples of the training set and 10000 examples of the test set, both of which are photographs from Zolando's articles. Each example is a 28*28 grayscale picture with a label from one of ten classifications.

5 Research Methodology

The goal of this research is to evaluate the performance of bespoke activation functions on various deep neural architectures such as CNN and DenseNet. Because these activation functions worked remarkably well on vanilla neural networks, it will be interesting to anticipate the model's loss and accuracy using these activation functions on deep neural architectures. The proposed system consists of the subsequent phases:

- phase 1: Determine the datasets on which the model must be trained.
- phase 2: Create train-test data partitions.

– phase 3: Perform one hot encoding on train and test datasets.

– phase 4: Create a predictive model with the CNN and DenseNet architectures.

– phase 5: Test the model and tune hyper-parameters.

– phase 6: Use categorical crossentropy as loss function and adam as optimizer.

– phase 7: Loss and accuracy value and plots.

Model Architecture: The model essentially recreates a network that functions similarly to neurons in our brain. The network is taught to make predictions based on previously collected data. In this study, we investigated the performance of various activation functions using deep neural networks such as CNN and DenseNet. Categorical CrossEntropy:A loss function utilised in classification problems is categorical crossEntropy.There are various objectives to complete, one of which is that an example can only belong to one of the potential categories.The model determines the category it is in here.

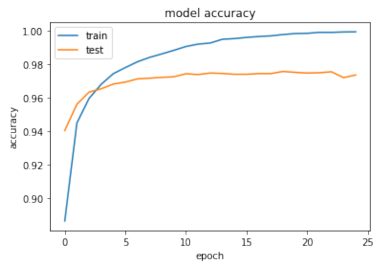
6 Performance Analysis

All of the following activation functions have been evaluated on various networks such as CNN and Denenet, as well as on various datasets such as MNIST, Cifar10, Fashion MNIST, and so on. We have a variety of activation functions, and it is difficult to select one that is appropriate for all test situations. Many aspects come into play, including whether or not it is differentiable, how quickly a neural network with a particular activation function converges, how smooth it is, whether it fits the constraints of the universal approximation theorem, and whether or not normalisation is retained.

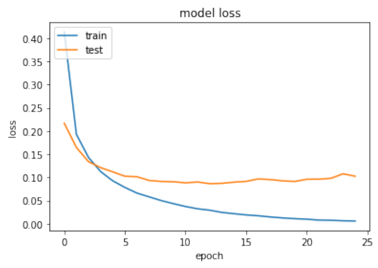
6.1 ReLU

case 1: ReLU was passed on cnn over Mnist dataset,accurated 0.9734 and had loss of 0.1025.

case 2: ReLU was passed on Densenet over cifar10 dataset,accuarted 0.8083 and had loss of 0.9868

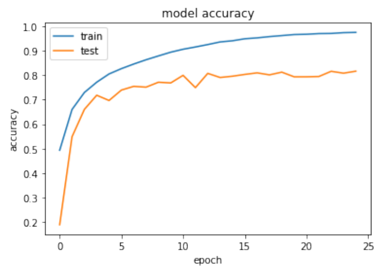


(a) CNN Mnist ReLU accuracy

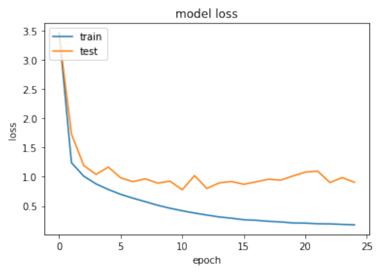


(b) CNN Mnist ReLU loss

Fig. 2: CNN ReLU



(a) Densenet cifar10 ReLU accuracy



(b) Densenet cifar10 ReLU loss

Fig. 3: Densenet cifar10

6.2 AReLU

AReLU has been tested on CNN using MNIST,cifar10, Fashion Mnist datasets. case 1: AReLU has best accuracy of 0.9734 on MNIST dataset and a loss of 0.0934 MNIST.

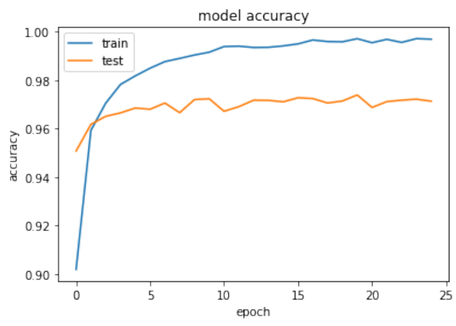
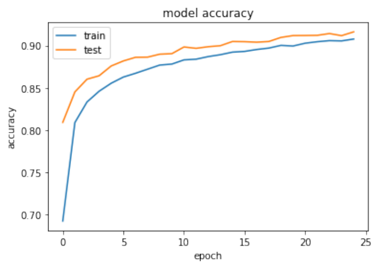
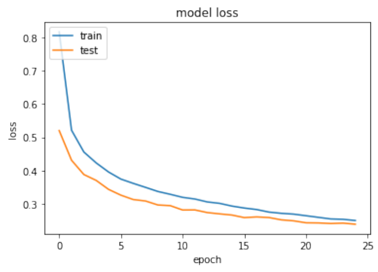


Fig. 4: CNN Mnist AReLU accuracy

case 2: AReLU accuated upto 0.9162 and a loss of 0.2386 on fashion MNIST dataset.



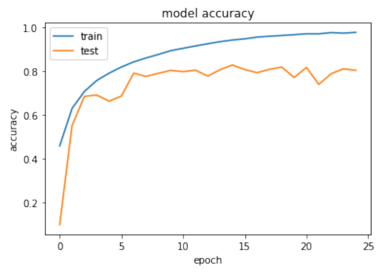
(a) CNN FashionMnist AReLU accuracy



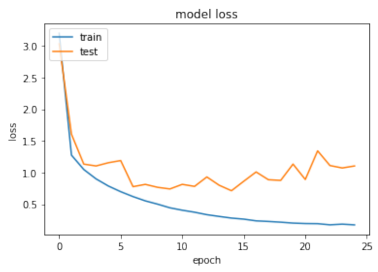
(b) CNN FashionMnist AReLU loss

Fig. 5: CNN FashionMnist AReLU

Case 3: AReLU was tested on Densenet over cifar10 dataset,accurate upto 0.7769 and had loss of 1.24



(a) DensNet cifar10 AReLU accuracy

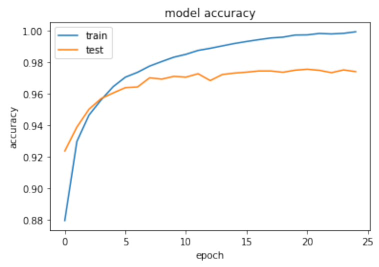


(b) DensNet cifar10 AReLU loss

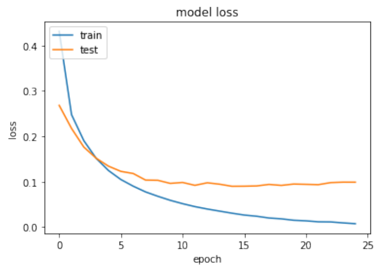
Fig. 6: DensNet cifar10 AReLU

6.3 ELU

ELU conducts resilient deep network training, resulting in higher classification precision. In comparison to other activation functions, ELU includes a saturation function for dealing with the negative section. When the unit is disabled, the activation function is reduced, causing ELU to execute quicker in the presence of noise. Case 1: ELU was evaluated on CNN over the Mnist dataset, and was accurate up to 0.9738 with a loss of 0.0988.



(a) CNN Mnist ELU accuracy

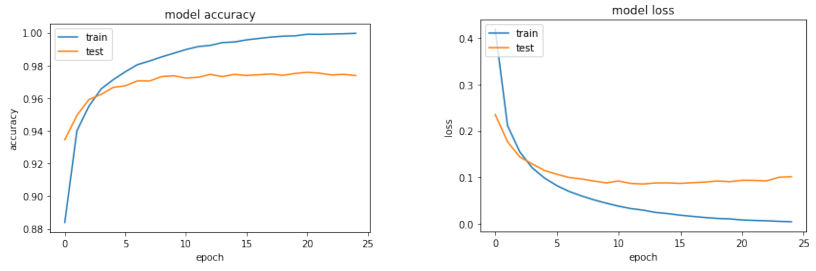


(b) CNN Mnist ELU loss

Fig. 7: CNN Mnist ELU

6.4 Mish

Mish activation function was passed on CNN over mnist dataset, accurate upto 0.9753 and had loss of 0.0934.



(a) CNN Mnist Mish accuracy (b) CNN Mnist Mish loss

Fig. 8: Mish Activation

7 Performance Summary

The purpose of this research is to demonstrate the performance of various activation functions and compare their performance across datasets and architectures. It may be summed up as follows: Because ReLU is not differentiable at X=0, the

	Architecture	**CNN**				**DenseNet**			
Performance Parameters	**Activation Function**	MNIST	CIFAR-10	CIFAR-100	FashionMNIST	MNIST	CIFAR-10	CIFAR-100	FashionMNIST
Accuracy	ReLU	0.9734	0.7809	0.481	0.9239	0.9939	0.9051	0.4762	0.9249
	AReLU	0.9746	0.7867	0.4686	0.9162	0.9927	0.8028	0.4607	0.9269
	Swish	0.9735	0.7612	0.4855	0.9191	0.9878	0.8164	0.4924	0.9273
	Mish	0.9738	0.7711	0.4535	0.9194	0.9934	0.8139	0.4781	0.9207
	ELU	0.9765	0.7761	0.4457	0.9164	0.3318	0.8152	0.5135	0.9113
	SELU	0.973	0.7492	0.3973	0.9031	0.9898	0.8061	0.4935	0.9251
	SBAF	NAN	NAN	NAN	NAN	NAN	NAN	NAN	NAN
	SBAF Parabola	0.9759	0.1	0.01	0.7229	0.1009	0.1	0.01	0.1
	Absolute Function	0.9754	0.7709	0.3363	0.8976	0.4994	0.5268	0.5193	0.8878
Loss	ReLU	0.1025	0.7136	2.0615	0.2155	0.0719	0.8169	2.6083	0.3698
	AReLU	0.2015	0.7615	2.062	0.2386	0.0686	1.1031	3.399	0.3656
	Swish	0.108	1.0686	2.4678	0.2332	0.1106	1.0618	3.2292	0.4038
	Mish	0.1013	1.0312	2.7021	0.2341	0.0793	1.1285	3.3057	0.5011
	ELU	0.0953	0.9142	2.7263	0.2417	0.0808	0.8714	2.6364	0.442
	SELU	0.1071	0.9346	2.6897	0.2729	0.104	0.8873	2.5875	0.3161
	SBAF	NAN	NAN	NAN	NAN	NAN	NAN	NAN	NAN
	SBAF Parabola	0.1654	2.5871	4.679	0.6913	3.0135	2.9098	5.0057	3.0092
	Absolute Function	0.1119	0.6761	2.5586	0.2825	2.6472	2.1403	2.4566	0.4565

Gradient will be 0. During Back-Propagation, we compute gradient as well as the gradient of Activation Functions. If the derivative equals zero, the gradient equals zero, there will be no weight update, and the network will finally stop learning. A-derivative ReLU's is not equal to 0 and is differentiable at x=0. A-ReLU is more tenacious than ReLU. A-ReLU was performing similarly to ReLU, and in some cases outperformed ReLU.

References

1. S. Saha, A. Mathur, K. Bora, S. Basak and S. Agrawal, ”A New Activation Function for Artificial Neural Net Based Habitability Classification,” 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 1781-1786, doi: 10.1109/ICACCI.2018.8554460.
2. Basak, S., Mathur, A., Theophilus, A.J. et al. Habitability classification of exoplanets: a machine learning insight. Eur. Phys. J. Spec. Top. 230, 2221–2251 (2021). https://doi.org/10.1140/epjs/s11734-021-00203-z
3. Mohapatra, Rohan et al. “AdaSwarm: Augmenting Gradient-Based Optimizers in Deep Learning With Swarm Intelligence.” the IEEE Transactions on Emerging Topics in Computational Intelligence; 10.1109/TETCI.2021.3083428(2021).
4. Yedida, Rahul and Snehanshu Saha. “Beginning with machine learning: a comprehensive primer.” The European Physical Journal Special Topics 230, 2363–2444 (2021). https://doi.org/10.1140/epjs/s11734-021-00209-7
5. Prashanth, Tejas et al. “LipGene: Lipschitz continuity guided adaptive learning rates for fast convergence on Microarray Expression Data Sets.” IEEE/ACM transactions on computational biology and bioinformatics; https://ieeexplore.ieee.org/document/9531348 (2021)
6. Saha, Snehanshu et al. “DiffAct: A Unifying Framework for Activation Functions.” International Joint Conference on Neural Networks (IJCNN) (2021); 1-8.
7. Mediratta, Ishita et al. “LipAReLU: AReLU Networks aided by Lipschitz Acceleration.” 2021 International Joint Conference on Neural Networks (IJCNN) (2021); 1-8.
8. Sarkar, Jyotirmoy et al. “An Efficient Use of Principal Component Analysis in Workload Characterization-A Study.” AASRI Procedia 8 (2014): 68-74.
9. Ravikiran, Manikandan et al. “TeamDL at SemEval-2018 Task 8: Cybersecurity Text Analysis using Convolutional Neural Network and Conditional Random Fields.” *SEMEVAL (2018).
10. Yedida, R., Saha, S. (2019). A novel adaptive learning rate scheduler for deep neural networks. ArXiv, abs/1902.07399.
11. Makhija, Simran et al. “Separating stars from quasars: Machine learning investigation using photometric data.” Astron. Comput. 29 (2019): 100313.
12. Sridhar, Shailesh et al. “Parsimonious Computing: A Minority Training Regime for Effective Prediction in Large Microarray Expression Data Sets.” 2020 International Joint Conference on Neural Networks (IJCNN) (2020): 1-8.
13. Ginge, Gouri et al., Mining massive databases for computation of scholastic indices: Model and quantify internationality and influence diffusion of peer-reviewed journals; Proceedings of the fourth national conference of Institute of Scientometrics, SIoT; 1-26 (2015)
14. Anisha R Yarlapati et al., Early prediction of LBW cases via minimum error rate classifier: A statistical machine learning approach; IEEE International Conference on Smart Computing (SMARTCOMP); 1-6 (2017)
15. S.Saha et al., DSRS: Estimation and forecasting of journal influence in the science and technology domain via a lightweight quantitative approach; Collnet Journal of Scientometrics and Information Management, 10(1), 41-70 (2016)
16. Saha, Snehanshu et al. “A New Activation Function for Artificial Neural Net Based Habitability Classification.” 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2018): 1781-1786.

17. Ramachandran, Prajit et al. “Swish: a Self-Gated Activation Function.” arXiv: Neural and Evolutionary Computing (2017): n. pag.
18. Safonova, Margarita et al. “Quantifying the classification of exoplanets: in search for the right habitability metric.” The European Physical Journal Special Topics (2021): Vol. 230, No. 10, pp. 2207–2220
19. Basak, Suryoday et al. “CEESA meets machine learning: A Constant Elasticity Earth Similarity Approach to habitability and classification of exoplanets.” Astron. Comput. 30 (2020): 100335.