A survey on recent activation functions with emphasis on oscillating activation functions

Bhavya Raitani
bhavya.raitani2021@vitstudent.ac.in

Abstract

The neural network learning and generalizing mechanisms of the brain have always intrigued scientists and researchers. Therefore, with the aim to mimic these "physiological workings" of the brain, the concept of using layers of artificial neurons (nodes) to make experienced predictions, accurate decisions, etc (Artificial neural network (ANN)) was introduced [5]. This paper briefly discusses few activation functions that have been used frequently over the years. In addition to that, the recent propositions regarding oscillating activation functions have been dealt with.

1 Introduction

A deep neural network uses representation learning models to train the system to learn the link between the given inputs and corresponding outputs in order to use this knowledge for predicting outputs for other similar inputs [12]. Each of these self-learning networks (neurons) comprise the input layer, multiple hidden layers and the output layer [3]. Since a group of linear functions may be condensed in the form of a single linear function, it is the activation function that adds non linearity to the functions in hidden layers, brings them to suitable ranges and induces biological similarities [11] [10]. The activation functions also include properties like continuity, differentiability (due to the need of gradient of loss function in backpropagation), and boundedness [2].

2 General working of a neuron

Each input is assigned with weights that are multiplied to the inputs $\sum_{i=0}^{n} W_i x_i$ [12] where x_i is the input of i^{th} layer and W_i is the weight assigned to it. To this product summation, biases are added and this resultant is passed on to the next step wherein a non-linear activation function is applied. Then the value from this function is passed as input to the next layer and the process continues until the output layer is reached. This process is called *forward propagation*. The output thus computed is juxtaposed with the actual value of output that is given.

Using the cost function $\sum_{i=0}^{n}(y - y')^2$ where y is the calculated result while y' is the expected correct result, the variation between actual and calculated value is obtained and the system tries to reduce that value to get closer to the actual output by adjusting the weights using optimizer algorithms like gradient descent, stoichastic gradient descent, etc [5]

$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$

where W_{new} is the new adjusted weight, W_{old} is the weight that was assigned in the previous forward propagation, η is the learning factor determined by hyperparameters, L is the calculated output [3]. This step is called *back propagation* and is the basis of neural network learning. [2]

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton also highlighted in their ‘Deep learning’ review paper, “The key aspect of deep learning is that these layers of features are not designed by human engineers: they *[the different functions like weights, biases, etc]* are learned from data using a general-purpose learning procedure.”[3]

3 General classification of activation functions:

3.1 Binary Step Function

$$H(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Also called the Heaviside function, this function is used in classifications as it gives binary output based on some predefined threshold value. Since it’s derivative is zero, this is not preferable when multiple cases are involved as there is no learning during the backpropagation steps.

3.2 Sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-}x}$$

The *“most widely used”* activation function, logistic sigmoid, condenses any function to its range of (0,1) while also binding the function from exploding. This function works well for function simulations as its derivative is smooth [14] and for classifications as it can give binary results.[10]

3.3 Tan h:

$$\tan h = \frac{1 - e^{-x}}{1 + e^{-x}}$$

As an enhancement to the sigmoid function by including negative outputs, it constructively deducts epochs for training comparatively and is primarily used for classifications [3]. Combating saturation of subsequent layer neurons while monitoring the movement of the mean of input, tanh improves CNN performance. [10]

3.4 Vanishing Gradient problem

In backpropagation during adjustment-readjustment of new weights, chain rule of derivatives is used in the formula. As layers increase, the derivative keeps on decreasing and the resultant product is almost negligible. Therefore, no appreciable changes to the new weight are done. This makes the overall procedure *“computationally expensive”* and slow.

3.5 ReLU:

$$f(x) = max(0, x)$$

In order to solve this Vanishing gradient problem and speed up the process compared to other activation functions, researchers explored the possibility of this function [14]. On one hand the pre-assignment of zeroes for negative inputs reduces the computation complexities, but inserts a non-differentiability element in the network whenever a negative bias is encountered and also shift the mean to positive bias [1]. Besides, its derivative might be small in certain conditions thereby not updating weights substantially.

3.6 Exploding Gradient problem

Though ReLU is extensively used in networks, it falls short of perfection by experiencing the exploding gradient problem. If the initial assigned weights are high, then by chain rule the consecutive product would be even greater. Hence, the difference between new weight and the previous weight would be large with the function failing to converge at the global minima.

3.7 Leaky ReLU:

$$y_i(x) = \begin{cases} x_i & x \geq 0 \\ \frac{x_i}{a_i} & x < 0 \end{cases}$$

To counter dead-neurons in ReLU, Maas et al., (2013) proposed leaky ReLU while suggesting the constant value to be large say 100 [6] This is used as a more flexible version of ReLU, though its constant derivative (here 0.01) might chance upon vanishing gradient again [15].

3.8 Softplus:

$$f(x) = \log(1 + e^x)$$

Introduced as an alternative to other s-shaped functions like sigmoid and tanh, softplus having sigmoid as its derivative along with efficacy in drop out regularizations, it successfully reduces vanishing gradient problem [16]. Their output, like sigmoid is between 0 and 1, gives the categorical probability distribution and are effective for Gaussian policy DNNs that require positive standard deviation due to non-zero gradient.

3.9 Swish:

$$f(x) = x.\sigma(\beta(x)) = x.sigmoid(\beta(x))$$

Diverging from the fixed activation functions, [Ramachandran et al., 2018] [13] introduced a trainable parameter that assessed performance of activation functions (candidate activation function), then training a “child network”, determining the best performing function for further manipulations. In light of its resemblance to ReLU, it is easy to deploy in networks previously working on ReLU. But this method of finding the appropriate function is computationally heavy. The facility to tune swish’s architecture in accordance with the network’s requirements incentivises researchers to exploit more of the trainable activation function capabilities. Gustineli (2022) also mentioned “Swish’s smooth, continuous profile was critical in deep neural network architectures for better information propagation when compared to ReLU.” [2]

3.10 Mish:

$$f(x) = x.\tan h.softplus(x)$$

Misra in his 2020 paper explained how the “first derivative behaviour” regularized and optimized the network [8]. Possessing characteristics like swish (non-monotonicity, smooth profile, self-gating property), mish matched or surpassed other generally accepted functions. The graph of mish contains a wide minima and a smooth output profile which helps in generalization.

4 Oscillating activation function

Jérémie Lefebvre et. al in their 2018 paper mentioned, “Oscillatory activity is a key component of brain dynamics.” Studying the statistics, they ascertained a dependence between nonlinearity and power of oscillations [4]. It is this dependence, that galvanised researches into more brain-like neuron exchanges involving oscillating functions. Utilizing its innate characteristic of positive and negative half-spaces, these functions are known to have the ability to give solutions with fewer neurons. So much, that the “linear nonseparable training example” [9] of XOR is proposed to have been solved using a single neuron by oscillating functions [10] [7].

4.1 GCU:

$$C(z) = z.cos(z)$$

Introducing “nonlinear decision boundaries” in activation functions, the 2021 paper presents a growing cosine unit (GCU) function [10]. In contrast to the common perception of a single plane decision boundary, the GCU has infinite evenly spaced solutions resulting in infinite evenly spaced hyperplanes. This property is utilized in the resolution of XOR. A simple single neuron solution to XOR problem using mean square loss, Stochastic Gradient Descent and signum may be obtained using GCU. When z tends to 0, it behaves as ReLU in positive region, for first maxima and minima it behaves like sigmoids and temporarily saturates while for large inputs it “oscillates and is an unbounded function” [10]. This proposed function is computationally light on the system as compared to Swish and mish (but not with respect to ReLU). This function improved gradient flow while consistently giving higher accuracy levels than other state-of-the-art functions (Sigmoids, Swish, Mish, ReLU) on CIFAR-10, CIFAR-100 and ImageNette architectures.

4.2 Other biologically inspired oscillating activation functions

The activation functions being utilized in DNNs these days are mostly monotonic in nature and unbounded with derivatives greater than or close to one. In the intensive research to find functions that have XOR property (functions that "can be used in a single neuron to learn the XOR function" [11]), it is imperative for the function to mimic identity function when x->0. Not only does this make an artificial neuron more brain-like but also is a requirement for expeditious training of neurons in the initial stages. Thus, functions need to be chosen with care and attention in order to not select a function that in itself would retard the efficiency of the network. The XOR function maybe written as

$$a \oplus b = D = \left\{ \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} \quad ,-1 \right), \ \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad ,1 \right), \ \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad ,1 \right), \ \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad ,-1 \right) \right\}$$

Mithra Noel et al.(2022)[11] proposed these functions in their paper:

- Shifted Quadratic Unit (SQU): $f(z) \ = \ z^2 + z$
- Non-Monotonic Cubic Unit (NCU): $f(z) \ = \ z \ - \ z^3$
- Shifted Sinc Unit (SSU): $f(z) \ = \ \pi \ \text{sinc}(z \ - \ \pi)$
- Decaying Sine Unit (DSU): $f(z) = \frac{\pi}{2}(\text{sinc}\,(z-\pi) - \text{sinc}\,(z+\pi))$

  where $\sin(z) = \begin{cases} 1 & z = 0 \\ \frac{\sin(z)}{z} & elsewhere \end{cases}$

All of these proposed functions are continuous, differentiable but not monotonic and for small values each of them resembles linear function. SQU and NCU possess 2 and 3 planes respectively while GCU and DSU have infinite making them conveniently capable of solving 2 hidden 1 output layer (XOR) problem with 1 neuron. These functions have faster convergence rates, require lesser training time per epoch while sporting high accuracy rates.

5 Conclusion

In hopes of lessening the gap between an actual and an artificial neuron, researchers have been trying to find most suitable functions. Through this paper different popular activation functions (sigmoids, tan h, ReLU, Softplus, Swish, Mish) have been discussed and compared. Additionally, the properties that make oscillating functions (like GCU) unique have been observed. The oscillating functions are proposed to have surpassed the previously popular activation functions by utilizing lesser neurons and system resources while also reducing the training time. This shows the incredible potential of oscillating activation functions in future.

References

[1] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural networks : the official journal of the International Neural Network Society*, 138:14–32, 2021.

[2] Murilo Gustineli. A survey on recently proposed activation functions for deep learning. *arXiv preprint arXiv:2204.02921*, 2022.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] Jérémie Lefebvre, Axel Hutt, Jean-François Knebel, Kevin Whittingstall, and Micah M Murray. Stimulus statistics shape oscillations in nonlinear recurrent neural networks. *Journal of Neuroscience*, 35(7):2895–2903, 2015.

[5] Ehsan Lotfi and M-R Akbarzadeh-T. A novel single neuron perceptron with universal approximation and xor computation properties. *Computational intelligence and neuroscience*, 2014, 2014.

[6] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[7] Ashutosh Mishra, Jaekwang Cha, and Shiho Kim. Single neuron for solving xor like nonlinear problems. *Computational Intelligence and Neuroscience*, 2022, 2022.

[8] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4(2):10–48550, 2019.

[9] Tom Mitchell. *Machine Learning*. McGraw-Hill Education, 1997.

[10] Mathew Mithra Noel, Advait Trivedi, Praneet Dutta, et al. Growing cosine unit: A novel oscillatory activation function that can speedup training and reduce parameters in convolutional neural networks. *arXiv preprint arXiv:2108.12943*, 2021.

[11] Matthew Mithra Noel, Shubham Bharadwaj, Venkataraman Muthiah-Nakarajan, Praneet Dutta, and Geraldine Bessie Amali. Biologically inspired oscillating activation functions can bridge the performance gap between biological and artificial neurons. *arXiv preprint arXiv:2111.04020*, 2021.

[12] Chigozie Nwankpa, Winifred L. Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378, 2018.

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

[14] Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. In *Bio-inspired neurocomputing*, pages 203–224. Springer, 2021.

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

[16] Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–4, 2015.