An Exploration of Alternative Sigmoidal and Rectified Linear Activation Functions

***Abstract*— We present two novel activation functions, named BAH and DReLU. BAH belongs to the sigmoidal family. DReLU is a member of the rectified linear family, and it has a periodic component with an optional learnable parameter. The properties of both of these activation functions are examined. Experimental results indicate that both functions perform comparably to baseline activation functions prevalent in neural network modelling for the tasks of classification on MNIST and CIFAR-10.**

***Index Terms*—activation functions, neural networks**

I. Introduction

An essential aspect of neural networks is the nonlinearity that is applied following linear operations. These nonlinear transformations are collectively referred to as activation functions. There are several such activation functions that have been proposed over the years. Moreover, different activation functions have been found to be particularly useful for specific problems or neural network architectures. For instance, the sigmoid activation function is commonly used as the final nonlinearity in binary classification problems, since the output can be interpreted as the probability of identifying the positive class.

Although many activation functions are described in the literature, there are only a few that are ubiquitously defined in machine learning libraries. The only true criteria qualifying a function as an activation function are continuity, a non-constant range, and differentiability over most of the domain (to facilitate backpropagation). The fact that there are such loose requirements greatly opens up the activation function discovery space. We decided to seize the opportunity and contribute to the investigation of novel activation functions.

We propose two activation functions: BAH and DReLU. The former is a sigmoidal activation function that morphologically resembles the hyperbolic tangent activation function, with the exception that its slope is slightly gentler. The latter is related to the rectified linear subset of activation functions and incorporates periodicity, along with an optionally learnable parameter.

This report chronicles the development and evaluation of the BAH and DReLU activation functions. First, we provide a brief survey of the literature regarding recently proposed activation functions. We then formally define the BAH and DReLU activation functions and elucidate their respective properties. What follows is an empirical evaluation of BAH and DReLU against commonly employed activation functions, such as ReLU. We then conclude by discussing the results and identifying directions for future work.

II. Related Works

There are many different activation functions that are used in neural networks. ReLU and the sigmoid function are among the most commonly used activation functions. It is necessary that an activation function is nonlinear for a neural network with at least two layers to be capable of approximating any real continuous function [1]. It is also advantageous for the activation function be continuously differentiable as we can then use the gradient to optimize the neural network. This is not a hard restriction; rather, it is a mathematically desirable property.

There are many functions that satisfy the desired conditions and it is worthwhile to explore different functions that can be used as an activation function. Elliott [2] proposes a new activation function called the "squash function", $\sigma_e(u) = \frac{u}{1+|u|}$. Squash is differentiable everywhere, having the derivative $\sigma_e'(u) = \frac{1}{(1+|u|)^2}$. The paper claims that the number of computations needed for this function is less than the number of computations needed for the exponential function which is used in the sigmoid function and it, therefore, is more computationally efficient. However, this paper was published in 1993 and it has not been tested on modern neural networks.

Swish is another activation function that was proposed in 2017. Replacing the ReLUs with Swish has increased the top-1 classification accuracy on ImageNet by 0.9% for Mobile NASNet-A and 0.6% for Inception-ResNet-v2 [3]. Swish is defined as $f(x) = x \cdot \mathrm{sigmoid}(x)$. It is worthwhile to note that Swish is a non-monotonic activation function. Although this function violates one of the perceived desirable properties of activation functions, it still performs just as well as ReLU. This paper claims that being unbounded above, bounded below, non-monotonic, and smooth are some of the reasons why Swish outperformed ReLU in their experiments. Being unbounded helps combat the vanishing gradient problem. Their experiments also show Swish outperforms ReLU in deep neural networks when the depth of the layers is between 40 and 50.

Mish is another recently proposed activation function [4]. It is defined as $f(x) = x \tanh(\mathrm{softplus}(x))$. This activation function was inspired by Swish and is also non-monotonic. Mish matches the performance of ReLU, leaky ReLU, and Swish across many computer image classification tasks. Their experiments show that Mish performed better than Swish and ReLU when the depth of the network is greater than 15 on the MNIST dataset. Their experiments also suggest that Mish is less sensitive to weight initialization than Swish.

Hendrycks and Gimpel introduced the Gaussian Error Linear Unit (GELU) [5]. This function is of the form $x\Phi(x)$ where $\Phi(x)$ is the cumulative distribution function of the standard normal. GELU matches ReLU on various tasks in image, text, and audio classification, demonstrating that it can be a viable alternative to ReLU.

Activation functions with learnable parameters have also been investigated [6]. The Adaptive Piecewise Linear (APL) function is one such function (see equation 1).

$$h_i(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s) \tag{1}$$

In the above equation, S is a predefined hyperparameter and a_i^s,b_i^s are parameters learned during the optimization step. Their experiments improved on the CIFAR-10 (7.51%), CIFAR-100 (30.83%) dataset and set a new benchmark on the Higgs Boson dataset from LHC [6].

III. Proposed Activation Functions

A. Activation Function Types

Activation functions are nonlinearities wedged between linear transformations in deep neural networks. Although there is a cornucopia of previously proposed activation functions, many appear to fall into two informal categories. First, *sigmoidal functions* tend to approach a greater or lesser asymptote as the input approaches positive or negative infinity respectively. Some examples include the logistic function, the hyperbolic tangent, and the Squash Function [2]. Second, *rectified linear functions* tend to approach infinity as the input approaches infinity, whereas they extend with a significantly shallower slope in the negative direction. The origin is typically the pivot point of the function. ReLU, Leaky ReLU, and Swish [3] fall into this category.

We propose two novel activation functions—one for each category. The BAH function is a sigmoidal activation function (full name redacted). The "Drunken Rectified Linear Unit" (DReLU) function belongs to the rectified linear category.

B. The BAH Function

The BAH activation function is defined for $x \in \mathbb{R}$ in equation 2

$$BAH(x) := \text{sgn}(x)(1 - e^{-\text{sgn}(x)x}) \tag{2}$$

where $\text{sgn}(x)$ is the sign function, which is defined for $x \in \mathbb{R}$ as in equation 3. The rightmost expression is used in the popular machine learning library PyTorch, which was used for our implementation.

$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} = \begin{cases} \frac{x}{|x|} & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases} \tag{3}$$

Figure 1 depicts a plot of BAH near the origin. Note that $\lim_{x\to\infty} BAH(x) = 1$ and $\lim_{x\to-\infty} BAH(x) = -1$.

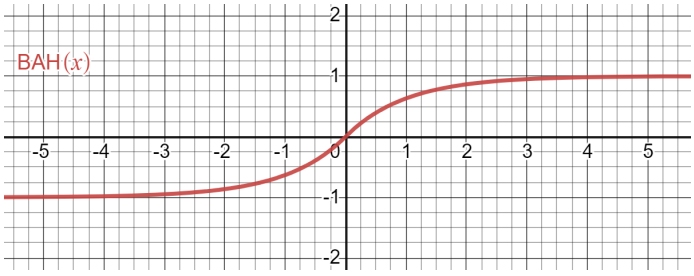

Fig. 1. The BAH Function. $BAH(x) := \text{sgn}(x)(1 - e^{-\text{sgn}(x)x})$

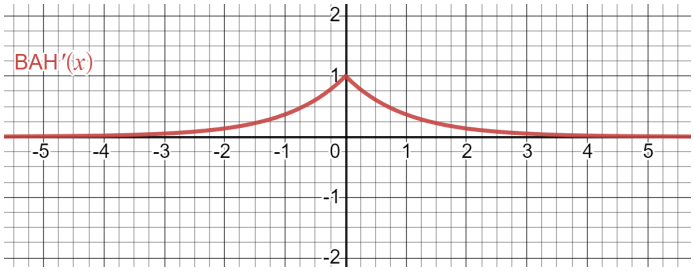

Fig. 2. The derivative of the BAH function, $BAH'(x) = 1 - \text{sgn}(x)BAH(x)$

Here we derive and discuss some properties of the BAH function. First, we note that BAH is continuously differentiable. Below we consider the derivative of the BAH function. We have to be careful with the sign function near the origin since $\text{sgn}(0) = 0$. First, we calculate the derivative of BAH $x \in \mathbb{R}$ such that $x \neq 0$, which is evaluated in equation 4.

$$\begin{aligned} \frac{d}{dx} BAH(x) &= \frac{d}{dx} \text{sgn}(x)(1 - e^{-\text{sgn}(x)x}) \\ &= (\text{sgn}(x))^2 e^{-\text{sgn}(x)x} \\ &= e^{-\text{sgn}(x)x} \end{aligned} \tag{4}$$

Now we consider the case when $x = 0$. Based on equation 4, we see that $\lim_{x\to 0^-} BAH'(x) = \lim_{x\to 0^+} BAH'(x) = 1$. We therefore establish that $BAH'(0) = 1$, thereby rendering BAH continuously differentiable. Figure 2 plots the derivative of BAH. We now note that the derivative of BAH can be expressed in terms of BAH itself, which is concisely expressed in equation 5.

$$\frac{d}{dx} BAH(x) = e^{-\text{sgn}(x)x} = 1 - \text{sgn}(x)BAH(x) \tag{5}$$

The fact that BAH is visible in its own derivative bodes well for backpropagation. It avoids having to recompute expensive exponential operations. We see this phenomenon in comparable sigmoidal activation functions, including the logistic function (i.e. $\sigma(x)$) and $tanh(x)$. It is widely known that $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$ and that $\frac{d}{dx} tanh(x) = 1 - tanh^2(x)$. With this in mind, table I compares the basic operations necessary to compute $BAH(x)$, $\sigma(x)$, and $tanh(x)$, along with their derivatives. The BAH function only requires a couple more operations than $\sigma(x)$ and $tanh(x)$, owing mostly to the usage of the sign function.

Function	+	-	*	/	exp
$BAH(x)$	0	1	3	1	1
$\sigma(x)$	1	0	1	1	1
$tanh(x)$	1	1	0	2	1
$BAH'(x)$	0	1	1	1	0
$\sigma'(x)$	0	1	1	0	0
$tanh'(x)$	0	1	1	0	0

TABLE I: Comparison of the minimum different operations for various sigmoidal activation functions. Derivative calculations assume that the function has already been computed. Recall that $\text{sgn}(x) = \frac{x}{|x|}$, requiring 1 division operation.

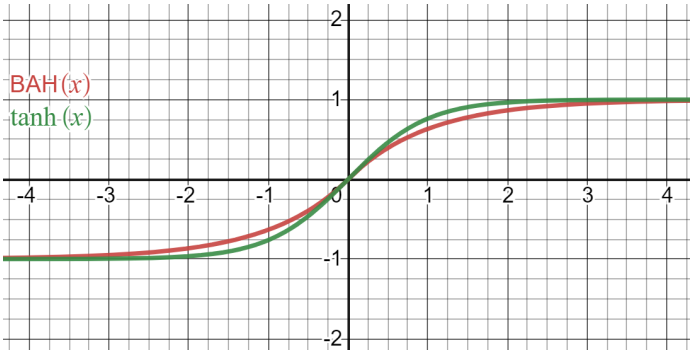

Fig. 3. A comparison of $BAH(x)$ (red) and $\tanh(x)$ (green).

Note that BAH is a strictly increasing function, since $BAH'(x) > 0$ for all $x \in \mathbb{R}$. It has a finite range of $(-1, 1)$, making gradient-based training methods stable but perhaps slower than for activation functions with infinite range.

Lastly, among popularly used activation functions, BAH is most similar to the hyperbolic tangent. Both have the same range and similar shape (see figure 3). Like the hyperbolic tangent function, the fact that BAH's range is centered around the origin prevents it from being intrinsically biased, like the logistic function (whose range is $(0, 1)$) [7]. The slope of BAH is generally less steep than that of the hyperbolic tangent near the origin and it approaches its horizontal asymptotes at more extreme values of x.

C. The DReLU Function

The DReLU activation function is defined for $x \in \mathbb{R}$ in equation 6.

$$DReLU(x) := \begin{cases} 0 & \text{if } x \leq 0 \\ x + \sin x & \text{if } x > 0 \end{cases} \tag{6}$$

The range of DReLU is $[0, \infty)$. As mentioned before, DReLU is a member of the rectified linear unit family of activation functions, which is also home to ReLU, Leaky ReLU, ELU, Softplus, Swish, and Mish. These functions are often applied following hidden layers in deep neural networks, rather than after output layers. They have unbounded ranges,

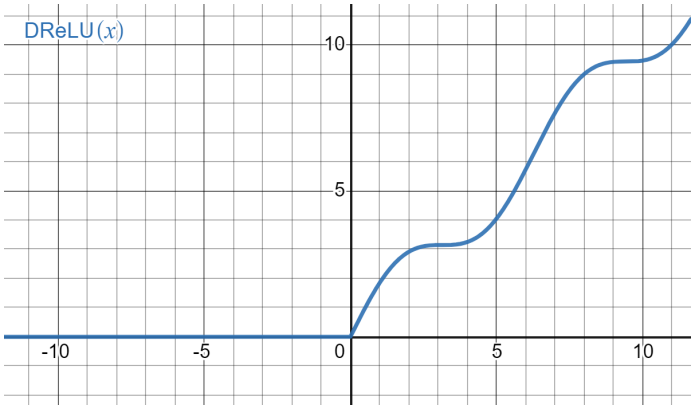

Fig. 4. The DReLU activation function, $DReLU(x) := \max(0, x + \sin x)$

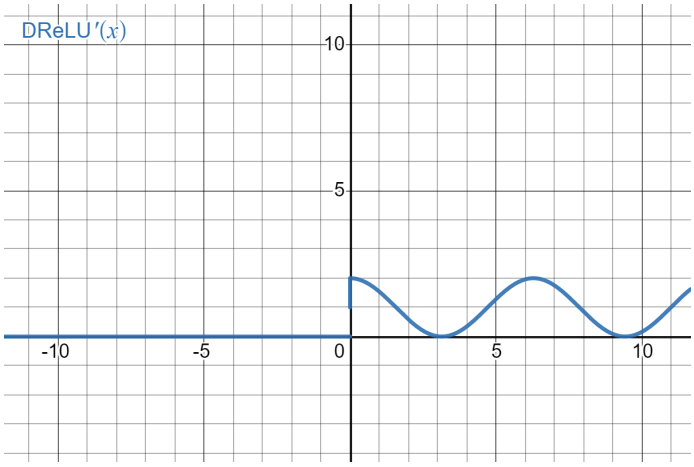

Fig. 5. The derivative of the DReLU function (see equation 7)

entailing large gradients, potentially facilitating efficient training. DReLU is very similar to ReLU, with the exception that a $\sin x$ term is added for the positive domain. Figure 4 portrays the DReLU function near the origin.

The derivative of DReLU is given in equation 7 and graphically portrayed in Figure 5.

$$\frac{d}{dx} DReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ \text{undefined} & \text{if } x = 0 \\ 1 + \cos x & \text{if } x > 0 \end{cases} \tag{7}$$

Note that, like ReLU, DReLU is neither smooth nor continuously differentiable. The derivative at the origin is undefined. In practicality, we set $DReLU'(x) = 0$. Unlike most rectified linear activation functions, the slope of the positive half of DReLU is periodically flat. These points correspond directly to the negatively sloped inflection points of the sine function. We believe that this would be an interesting activation to study because it enables multiple opportunities for potential convergence at flat regions of the function. It is possible that this phenomenon may be detrimental to neural network optimization in many cases, but it is nonetheless a worthwhile

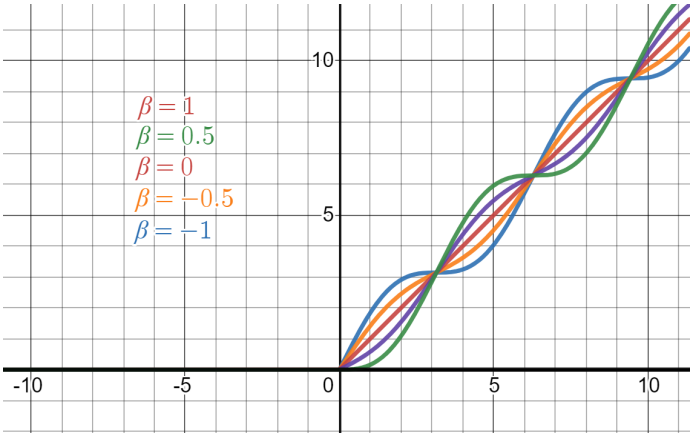

Fig. 6. The parameterized DReLU plotted for multiple values of β

problem to study. It could be that some learning problems benefit from periodically appearing flatter regions.

DReLU, as defined in equation 6, is monotonically increasing. Perhaps we desire an activation function that has a periodically gentle slope instead of being periodically flat. If we multiply the sine term in DReLU by a hyperparameter $\beta \in \mathbb{R}$, we see that the flat regions become inclined. Equations 8 and 9 give expressions for this "parameterized DReLU" and its derivative respectively.

$$DReLU(x) := \begin{cases} 0 & \text{if } x \leq 0 \\ x + \beta \sin x & \text{if } x > 0 \end{cases} \tag{8}$$

$$\frac{d}{dx} DReLU(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 + \beta \cos x & \text{if } x > 0 \end{cases} \tag{9}$$

When $\beta = 1$, the function is equivalent to the definition from 6. When $\beta = 0$, the function reduces to ReLU. Observe that if $\beta = -1$, DReLU is smooth and continuously differentiable. Figure 6 demonstrates the effect on DReLU of modifying the value of β within the range $[-1, 1]$. Values for β outside of $[-1, 1]$ break the monotonicity of the function.

The hyperparameter β can either be preset or made to be a learnable parameter. In this work, we investigate the cases where $\beta = 1$ and when β is a learnable parameter. In the latter scenario, we will report the value of β for each DReLU instance after the final training epoch. The initial value for β is sampled from the uniform distribution $U(0, 1)$.

IV. Evaluation

We completed multiple experiments to compare BAH and DReLU against the following commonly used activation functions: ReLu, Tanh, sigmoid, and SiLU (i.e. Swish). The evaluation first examines the functional runtime for each activation function. Next, we present results of training standard neural network architectures to perform benchmark classification tasks, replacing the activation functions of the hidden layers.

All experiments were implemented in Python 3, using PyTorch. Note that the activation functions were implemented as PyTorch modules. Code for all experiments available via our Colab Notebook[1].

A. Runtime Analysis

We begin the evaluation of our proposed activation by comparing the runtime of the proposed and standard activation functions identified above. To compare the runtime, a simple, measurable training task was defined and executed for each activation function considered. A small neural network was defined with a 1000-node hidden layer and single node output layer. The model was trained for 1000 epochs on a random dataset $D \in \mathbb{R}^{1000\times100} \sim \mathcal{N}(0, 1)$ with targets $t \in \mathbb{R}^{1000\times1} \sim \mathcal{U}(0, 1)$ to minimize the mean squared error loss. This experiment involves running the activation functions forward and backward; therefore, it is a fair way to evaluate runtime. Ten training trials were conducted for each activation function. The average execution time of this task for each activation function is reported in Table II.

Activation Function	Time [s]
BAH	1.0719
ReLU	0.7840
TanH	0.7710
DReLU	0.8920
Parameterized DReLU	0.9921
Sigmoid	0.7679
SiLU	0.7625

TABLE II: Average time taken to train a small arbitrary network on a randomly sampled dataset. Ten trials were completed for each network.

It is evident from the table that the newly proposed activation functions run more slowly than the standard activation functions. All the task executions employing the standard activation functions finished in under 0.8 sec. It is worthy to remark that the execution time of the task with regular DReLU (i.e. $\beta = 1$) is only slightly greater. We defer our speculation as to why the proposed functions were slower to Section V.

B. Benchmark Tasks

Our proposed activation functions are only as beneficial as they are good for standard machine learning tasks. To that end, we trained standard neural network architectures for a couple of benchmark classification tasks, using different choices for the activation functions of hidden layers.

We first trained a neural network with the LeNet architecture [8] to classify handwritten digits from the MNIST dataset. Training, validation, and test sets of 54000, 6000, and 10000 images were created. The model was trained for 50 epochs to minimize the categorical cross-entropy loss function. This training experiment was repeated once for each of our proposed activation functions and the standard activation functions. Refer to Figure 7 and Figure 8 for plots showing the

[1]Code is available at https://tinyurl.com/drelu-bah

training loss and validation accuracy plots over all epochs for each trial. Table III contains the results of these trials in terms of test set performance. DReLU achieves the best test accuracy of 0.9906 on the MNIST dataset, whereas the Tanh activation function achieves the lowest average categorical cross-entropy test loss of 0.0264.

Activation Function	Test Accuracy	Test Loss
BAH	0.9903	0.0281
ReLU	0.9887	0.0392
Tanh	0.9904	**0.0264**
DReLU	**0.9906**	0.0318
Parameterized DReLU	0.9894	0.0404
Sigmoid	0.9535	0.1523
SiLU	0.9871	0.0592

TABLE III: LeNet on MNIST dataset

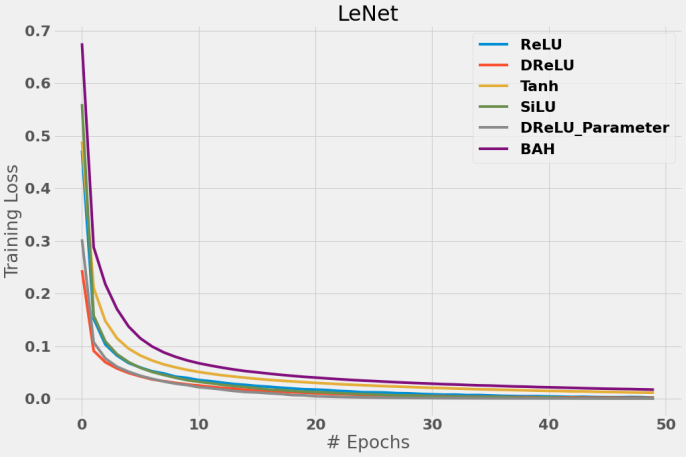

Fig. 7. Categorical cross entropy loss on the training set for multiple LeNet models trained on MNIST, each using a different activation function between the hidden layers. The trial with the sigmoid function is not pictured, as its training loss was significantly greater than in the other trials.

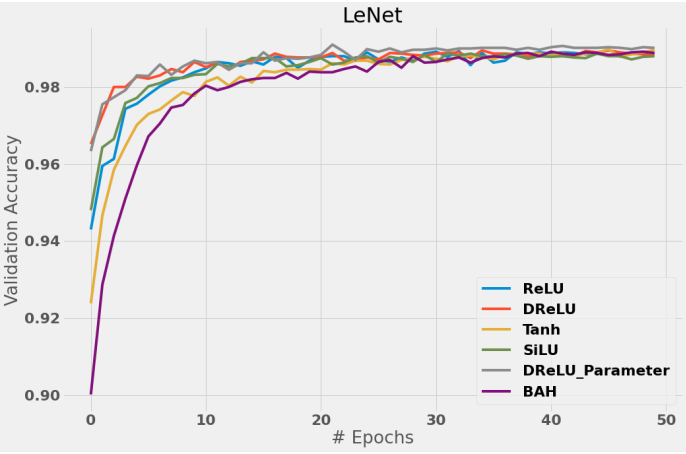

Fig. 8. Accuracy on the validation set for multiple LeNet models, each using a different activation function between the hidden layers. The trial with the sigmoid function is not pictured, as its validation accuracy was significantly lower than in the other trials.

Activation Function	Accuracy	Test Loss
BAH	0.7839	1.2875
ReLU	0.8339	0.9598
Tanh	0.7825	1.3258
DReLU	0.8294	**0.8943**
Parameterized DReLU	0.8226	0.9199
Sigmoid	0.7780	1.2949
SiLU	**0.8415**	0.9490

TABLE IV: VGG11 on CIFAR-10 dataset

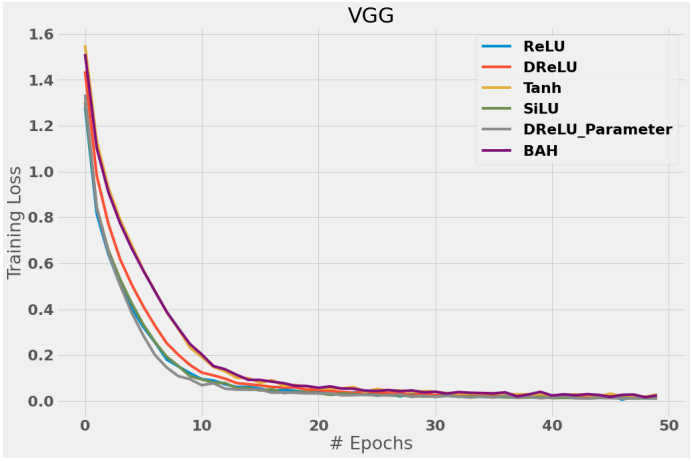

Fig. 9. Categorical cross entropy loss on the training set for multiple VGG11 models trained on CIFAR-10, each using a different activation function between the hidden layers. The trial with the sigmoid function is not pictured, as its training loss was significantly greater than in the other trials.

The second benchmark task was to train a neural network with the VGG11 architecture [9] to classify images from the CIFAR-10 dataset [10]. Once again, training, validation, and test sets of 45000, 5000, and 10000 images were created. The model was trained for 50 epochs to minimize the categorical cross-entropy loss function. As before, this procedure was repeated once each for BAH, DReLU, and the standard activation functions. Figure 9 shows the change in training set loss for all models trained and Figure 10 depicts the increase in accuracy on the validation set. The performance of these models on the test set is provided in Table IV. The SiLU (Swish) activation function achieves the best accuracy of 0.8415 and DReLU achieves the lowest test loss of 0.8943.

Finally, we documented the final values for β in each instance of the parameterized DReLU layer. Recall that in these experiments, β was a learnable parameter initialized randomly from a uniform distribution: $\beta \sim \mathcal{U}(0, 1)$. Table V reports the values of β learned after the training experiments focused on the parameterized DReLU. We observed that β converged to values in the range $[0, 1]$ in LeNet; however, this was not the case for VGG11.

V. Discussion

Our experiments indicated that both the proposed activation functions work as intended. BAH belongs to the same family

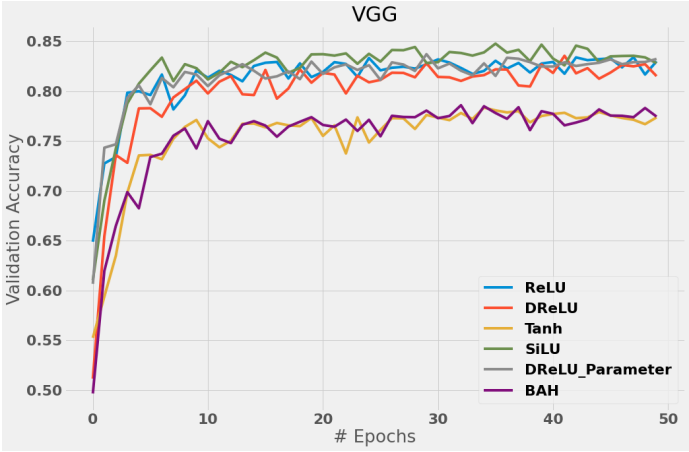

Fig. 10. Accuracy on the validation set for multiple VGG11 models, each using a different activation function between the hidden layers. The trial with the sigmoid function is not pictured, as its validation accuracy was significantly lower than in the other trials.

DReLU Layer	Final β (LeNet)	Final β (VGG11)
1	0.8512	1.1081
2	0.7229	1.1111
3	0.7494	1.4710
4	0.3328	−1.2939
5	N/A	0.3674
6	N/A	1.5306
7	N/A	0.3216
8	N/A	−1.0657

TABLE V: The values of β for each parameterized DReLU activation layer, after training LeNet on MNIST and VGG11 on CIFAR-10 for 50 epochs.

as the sigmoid and hyperbolic tangent activation functions, and it seems to outperform the sigmoid function in the intermediate layers. Moreover, BAH performed comparably to the hyperbolic tangent function. This is unsurprising since both functions share the same range and have nearly the same rate of change over their domains. DReLU and the parameterized generalization of DReLU are comparable to ReLU and SiLU in terms of performance, as evident in the experimental results.

It is interesting to note that our DReLU family of activation functions is unconventional in its design and contravenes some of the traditionally desirable features of activation functions, mainly owing to its periodic flat regions. However, our experimental results did not reveal any flagrant consequences due to this nature. We observed that, when training with the parameterized version of DReLU, β did not converge to 0. In that case, it would have reduced to ReLU. It could be that the gradients of the weights/biases of the network are robust to the flat and/or decreasing regions of the activation function induced by the periodicity of the sinusoidal term. Perhaps there is even something beneficial about the sinusoidal nature of the activation function. It is also worthwhile to note that $|\beta| > 1$ in some parameterized DReLU layers of the trained VGG11 network. In these cases, the activation function is no longer monotonically increasing; rather, it is convex over certain intervals. This was a striking result.

Lastly, the runtime experiments indicated that the proposed activation functions are slightly slower than the pre-existing activation functions in the PyTorch library. One possible explanation for this is that PyTorch activation functions are implemented using C/C++ and have a separate CUDA implementation if a GPU is being used. Our implementation did not take advantage of such acceleration. The benchmarks for runtime could be more accurate if such an implementation were to be carried out since the standard activation functions available in PyTorch are already optimized for CUDA.

VI. Conclusion

The present work involved the proposal and evaluation of two novel activation functions. The BAH function is a sigmoidal monotonically increasing function with range $(-1, 1)$. DReLU, on the other hand, is a rectified linear monotonically increasing function with range $[0, \infty)$ whose hallmark is periodic regions with gentler slopes. It has a hyperparameter β that controls the slope of the flatter regions that is optionally learnable.

Our results indicated that DReLU performed very well when applied in the LeNet neural network on the MNIST dataset, obtaining a test accuracy of 99.06%. However, several of the standard activation functions performed comparably; it is difficult to draw any conclusions as to which is definitely greatest. BAH performed very well, attaining a test accuracy of 99.03% for the same task. DReLU achieved the lowest test loss on the CIFAR-10 dataset when used with VGG11. DReLU also had the third-highest test accuracy of 82.94% and BAH achieved a slightly lower test accuracy of 78.39%.

There exist multiple possibilities for future work. First, it is possible that further performance gains could be realized by developing CUDA implementations of BAH and DReLU. Since several neural network training loops are offloaded to GPUs for accelerated parallelized computation, practical implementations of our activation functions should be CUDA-accelerated. Second, we can experiment with a wider range of neural network architectures that will give us a better comparison of the different activation functions. Finally, it would be worthwhile to investigate the reason as to why the β parameter does not converge to 0, which would reduce DReLU to ReLU. Furthermore, such work could seek to answer why β occasionally converges to values that break the monotonicity of DReLU.

Acknowledgment

We wish to express deep gratitude for our beloved professor, Dr. Orchard. His fatherly jokes kept us laughing through these trying times. ☺

References

[1] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, p. 303–314, 1989.

[2] D. Elliott, “A Better Activation Function for Artificial Neural Networks,” Tech. Rep., 1993.
[3] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” *CoRR*, vol. abs/1710.05941, 2017. [Online]. Available: http://arxiv.org/abs/1710.05941
[4] D. Misra, “Mish: A self regularized non-monotonic activation function,” 2020.
[5] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” 2020.
[6] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, “Learning activation functions to improve deep neural networks,” 2015.
[7] T. Yang, Y. Wei, Z. Tu, H. Zeng, M. A. Kinsy, N. Zheng, and P. Ren, “Design space exploration of neural network activation function circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1974–1978, 2019.
[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, pp. 541–551, 1989.
[9] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556
[10] A. Krizhevsky and G. Hinton, “Learning Multiple Layers of Features from Tiny Images,” University of Toronto, Toronto, Tech. Rep., 2009.