SINUSOIDAL ACTIVATION FUNCTIONS

An empirical study of a toy dataset.

Yuval Shechter
Math 597f - Fourier Methods - Final Project

1 Introduction

Neural Networks are commonly used in many different modern applications that require the use of a function of an unknown form but a known, finite amount of true input and output data drawn from the true distribution. They do this - in their most simplified structure - by using a structure like the following:

$$\vec{x}_k = \sigma(W_{k-1}^T \vec{x}_{k-1} + \vec{b}_{k-1}) \tag{1}$$

In this structure x_0 would be the input to this unknown "black box" function, some x_n for some arbitrarily chosen n would be considered the output, and W and $\vec{b}$ are both some matrices containing adjustable weights.
By using a technique known as backpropogation, the W_i's and $\vec{b}_i$'s can be adjusted to more closely approximate the distribution from which the supplied data points are drawn from.[1]
The one reminaing part of this network are the σ's which are the "non-linearities" or "activation functions" and will be the main focus of this writeup.

2 Activation Functions

Activation functions are a specific type of function that contain a non-linearity (ie; they are are not a linear transformation of the identity function $f(x) = x$), commonly used examples are *ReLU*: $\sigma = max(0, x)$, *Sigmoid*: $\sigma = (1 + \exp -x)^{-1}$, *Tanh*: $\sigma = tanh(x)$, and a variety of others. [2] Not as commonly used, however, are periodic activation functions. Called "Fourier Neural Networks" neural networks that use activation functions such as $cos(x)$ and $sin(x)$ have been proposed and used in previously existing literature to a varying degree of success. [3] [4] In this writeup there will be an exploration and analysis of their effectiveness on a commonly used benchmark task.

3 Methodology

3.1 Dataset

The benchmark task for testing this kind of activation function used is the *MNIST* dataset which is composed of a number of greyscale, handwritten digits (0-9) for which the task is to map from the greyscale image to the number it represents. [5]

3.2 Network Architecture

The neural network is composed of three layers (four iterations of k from 1). Where the dimensions are as follows:

1. x_0 is of size 784 (input image)
2. x_1 is of size 512
3. x_2 is of size 512
4. x_3 is of size 10 (output number)

Between each of the layers there is an activation function, and the output format is a one-hot vector ($\vec{0}$ with a 1 in the place of the predicted number). [6]

3.3 Procedure

In order to evaluate the effectiveness of sinusoidal activation functions, the procedure was as follows:

1. Create a hyperparameter sweep over learning rates, optimizers, batch sizes, and - of course - activation functions.

2. Train each network using a set of hyperparameters on a train, test split of the dataset.
3. Evaluate the results of this sweep using metrics such as training loss, testing loss, accuracy, precision, recall, and F1-score.

The hyperparameter sweep included learning rates of $10^{-i}; i \in [2, 4]$, the optimizers: *SGD* and *Adam*, batch sizes: $[16, 32, 64]$, and activation functions: *ReLU* (as a baseline); *Sin*; *Cos*; *Triangular Wave.*
The loss function used was *BinaryCrossEntropy* and the number of epochs was 20.

4 Results

4.1 Average Performance Comparison

The results ended up being fairly expected, the baseline *ReLU* activation function performed significantly better than the other activation functions analyzed:

Metric	ReLU	Triangular Wave	Sin	Cos
F1	0.9208	0.7773	0.7872	0.6709
Accuracy	0.9202	0.7759	0.7848	0.6819
Precision	0.9183	0.7745	0.7834	0.6789
Recall	0.9150	0.7676	0.7814	0.6637

Table 1: Mean Final Epoch Metric Across Sweep.

Metric	ReLU	Triangular Wave	Sin	Cos
F1	0.1008	0.3153	0.3115	0.3901
Accuracy	0.1031	0.3147	0.3117	0.3756
Precision	0.1065	0.3142	0.3117	0.3785
Recall	0.1146	0.3232	0.3121	0.3983

Table 2: Standard Deviation of Final Epoch Metric Across Sweep.

The ReLU activation function worked significantly better and was more consistent than any of the sinusoidal functions, with a mean higher by about 20% and a standard deviation lower by around the same amount than all of the other candidates. It is also easy to see that while the *Triangular Wave* and *Sin* had similar performance, *Cos* consistently underperformed both of them.

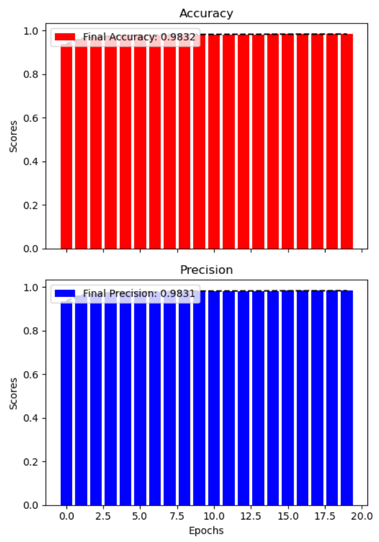
4.2 Best Performance Comparison

Metric	ReLU	Triangular Wave	Sin	Cos
F1	0.9827	0.9830	0.9812	0.9827
Accuracy	0.9828	0.9832	0.9813	0.9828
Precision	0.9826	0.9831	0.9811	0.9827
Recall	0.9826	0.9831	0.9811	0.9827

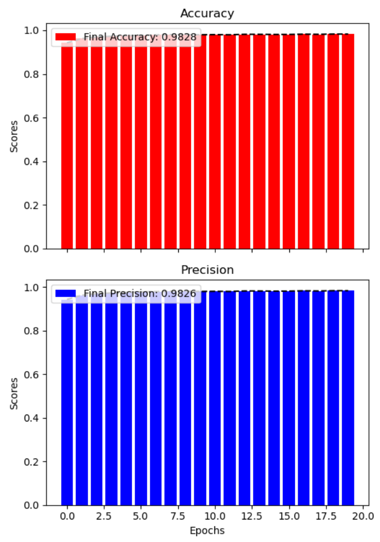
Table 3: Best performance comparison.

Unlike the average performance, the best performance at the end of training across the board is achieved by the *Triangular Wave* and with a higher learning rate of 10^{-2} and the *SGD* optimizer instead of *Adam.* This suggests that the gradients using this activation function are somehow more robust than those using the other activation function types. [7]

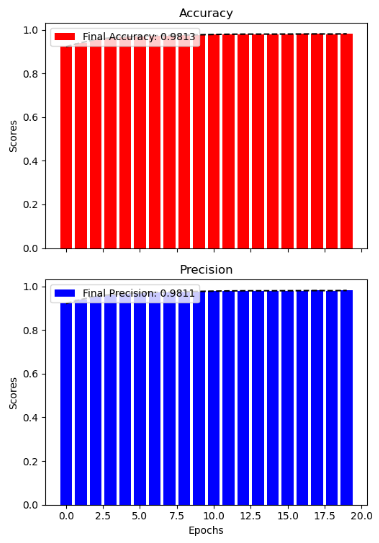
4.3 Best Performance Graphics



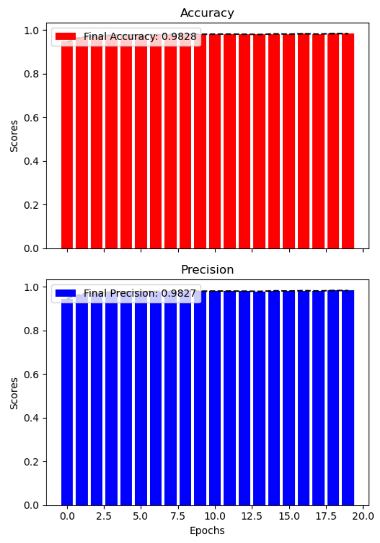
(a) Triangular Wave: LR: 10^{-2}, Optimizer: SGD, Batch Size: 16.



(b) ReLU: LR: 10^{-4}, Optimizer: Adam, Batch Size: 16.



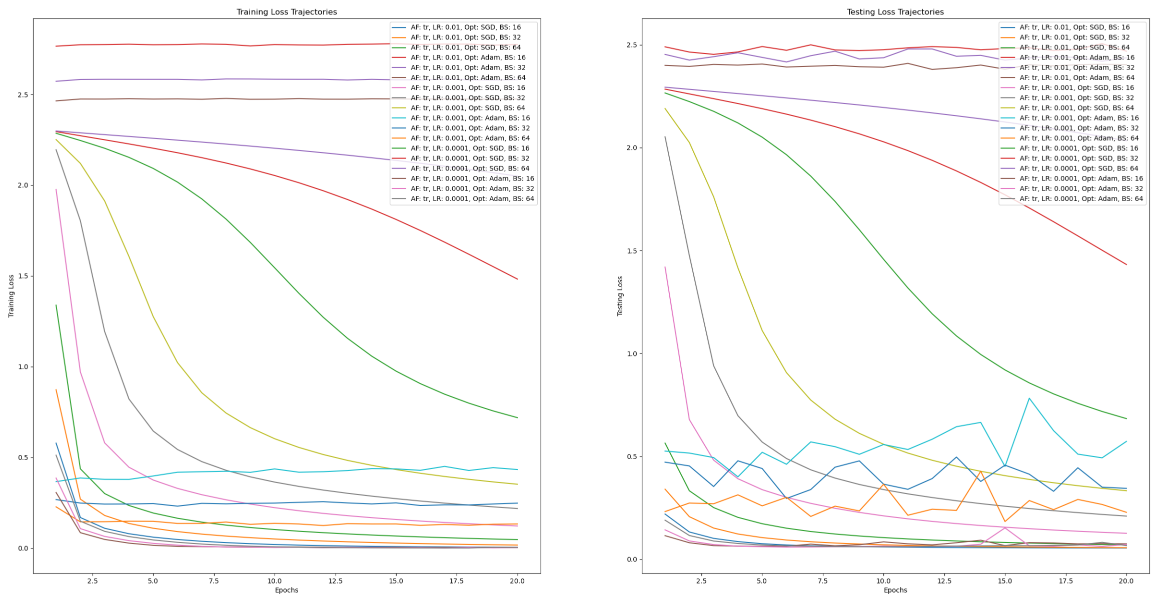
(c) Sin: LR: 10^{-4}, Optimizer: Adam, Batch Size: 32.



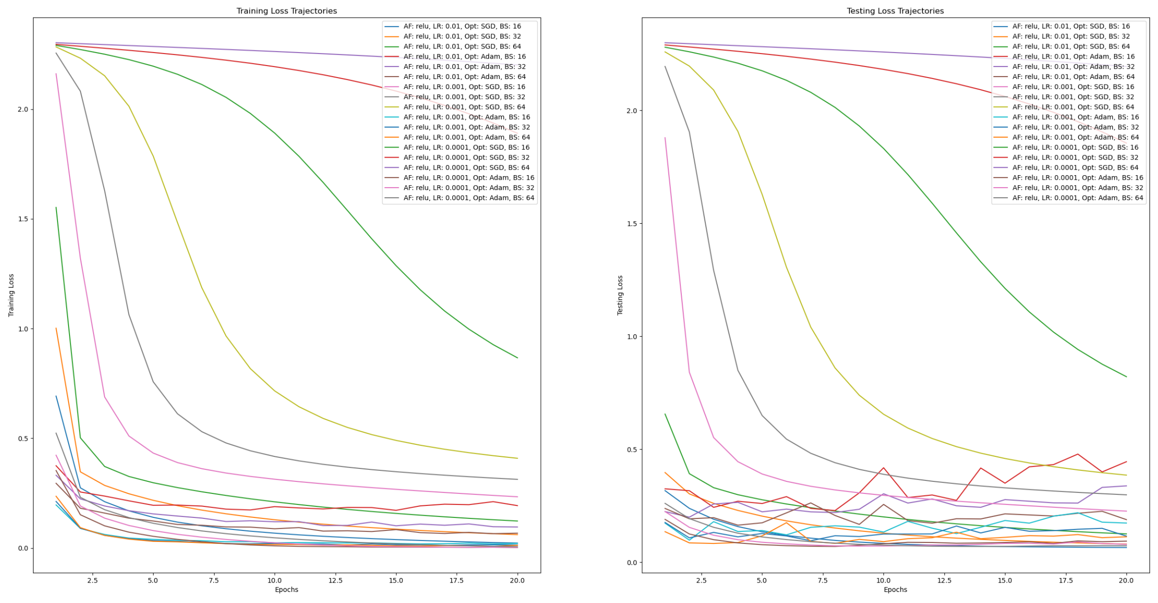
(d) Cos: LR: 10^{-4}, Optimizer: Adam, Batch Size: 16.

Figure 1: Bar Graphs of Best Performing Hyperparameters for Each Activation Function. Clearly, they are characterized by a similar behavior of getting high accuracy immediately and improving marginally with later epochs.

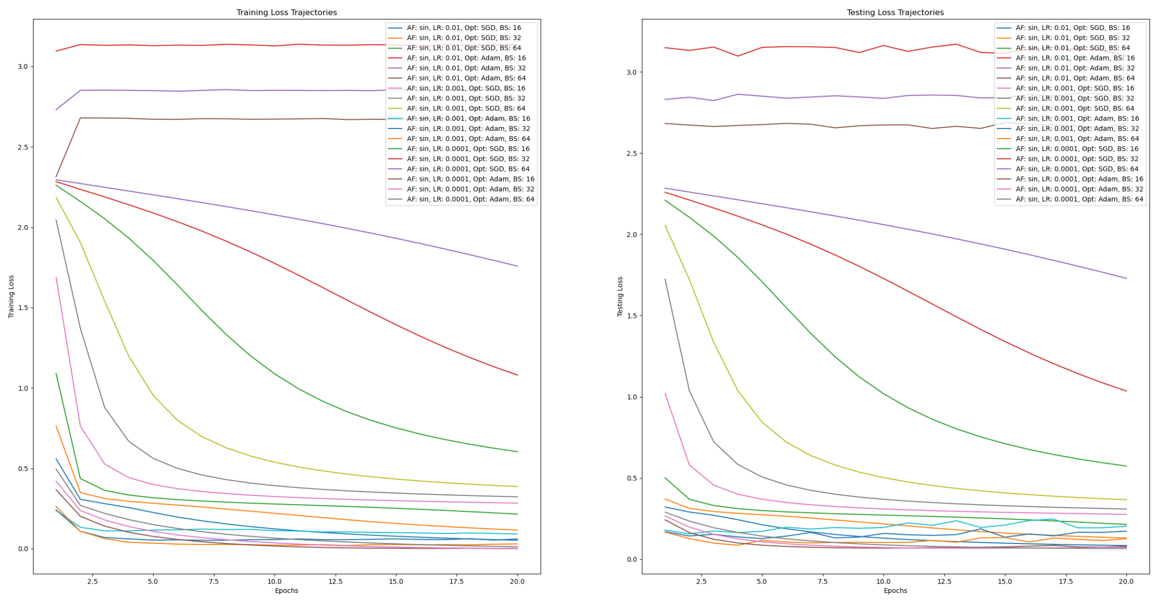
4.4 Performance Comparison by Activation Function



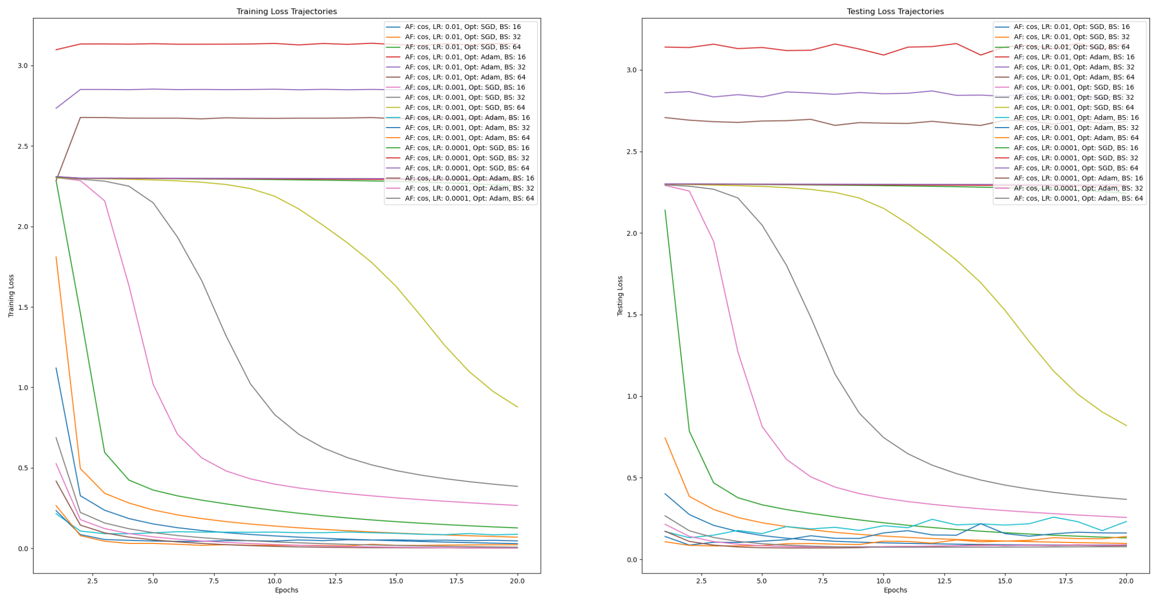
(a) All hyperparameters trajectories for Triangular Wave activation function.



(b) All hyperparameters trajectories for ReLU activation function.



(c) All hyperparameters trajectories for Sin activation function.



(d) All hyperparameters trajectories for Cos activation function.

Figure 2: Graphs of all the training and testing losses for each hyperparameter for each activation function.

4.5 Triangular Wave Best Performance Activation Range

A key part of using a periodic activation function (as opposed to an aperiodic one) is that intuitively its periodic nature should allow for the network to use a larger range of inputs to be squeezed between $[0, 1]$.

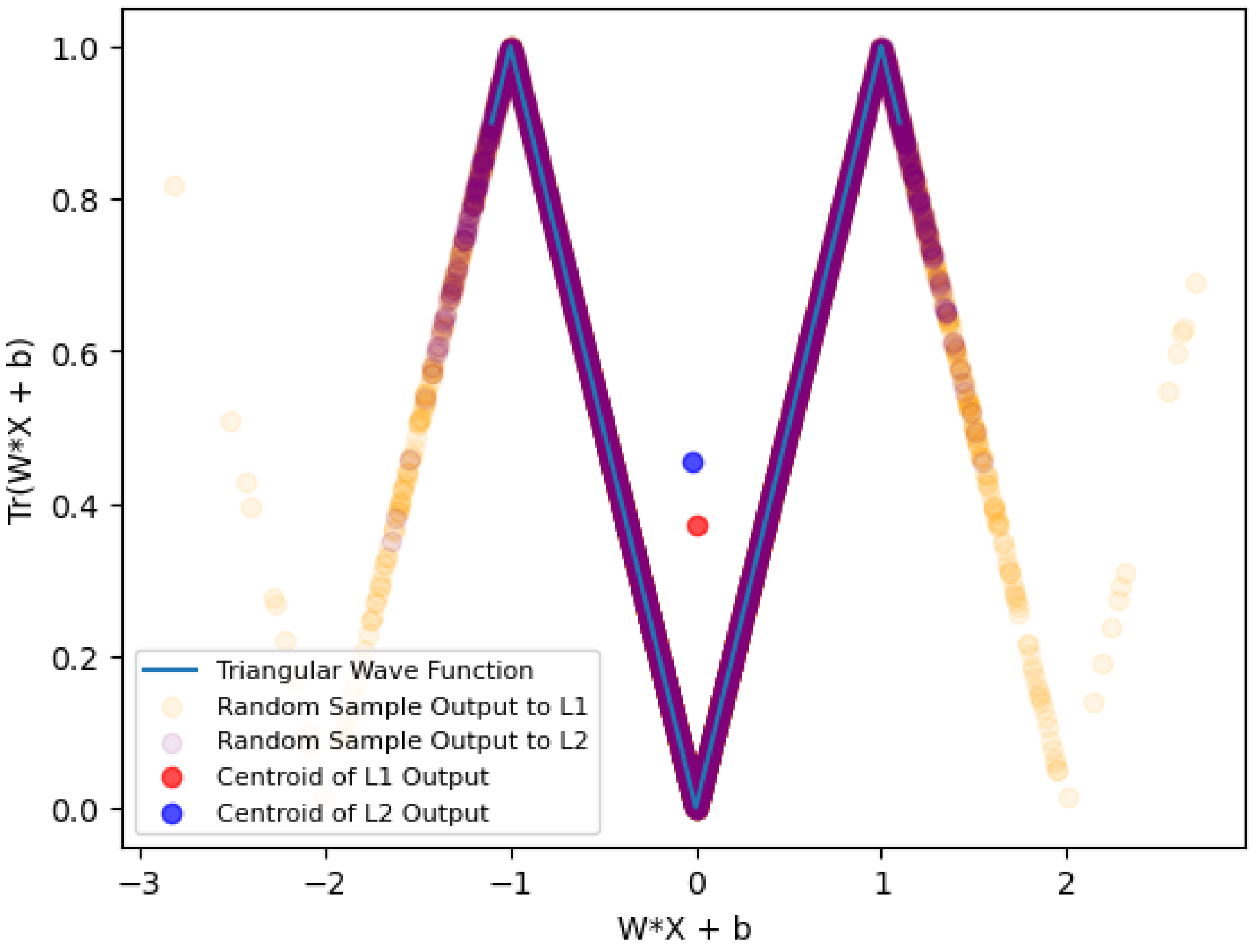


Figure 3: This graphic shows a sample of 10,000 points drawn from the test set and plots the distribution of the inner linear component operation (the input to the *Triangular Wave* activation function) against the output of the activation function on the same point.

We see however that the network doesn't seem to continue to use this periodic nature between the first and second layers as much. It's plain to tell that the distribution of those same points that were inputs to the first layer (L1) are contracted into a much smaller domain distribution for the second layer (L2). This is not as obvious as it might initially seem since the input to L2 is of the form $W_i\vec{X} + \vec{b}_i$ with W being an adjustable parameter, so even though $L1(x) \in [0, 1]$, $W_2 * L1(x) + b_2$ is not constrained to that same range. This implies that with more layers, the range may continue to constrain to only the domain $[0, 1]$.

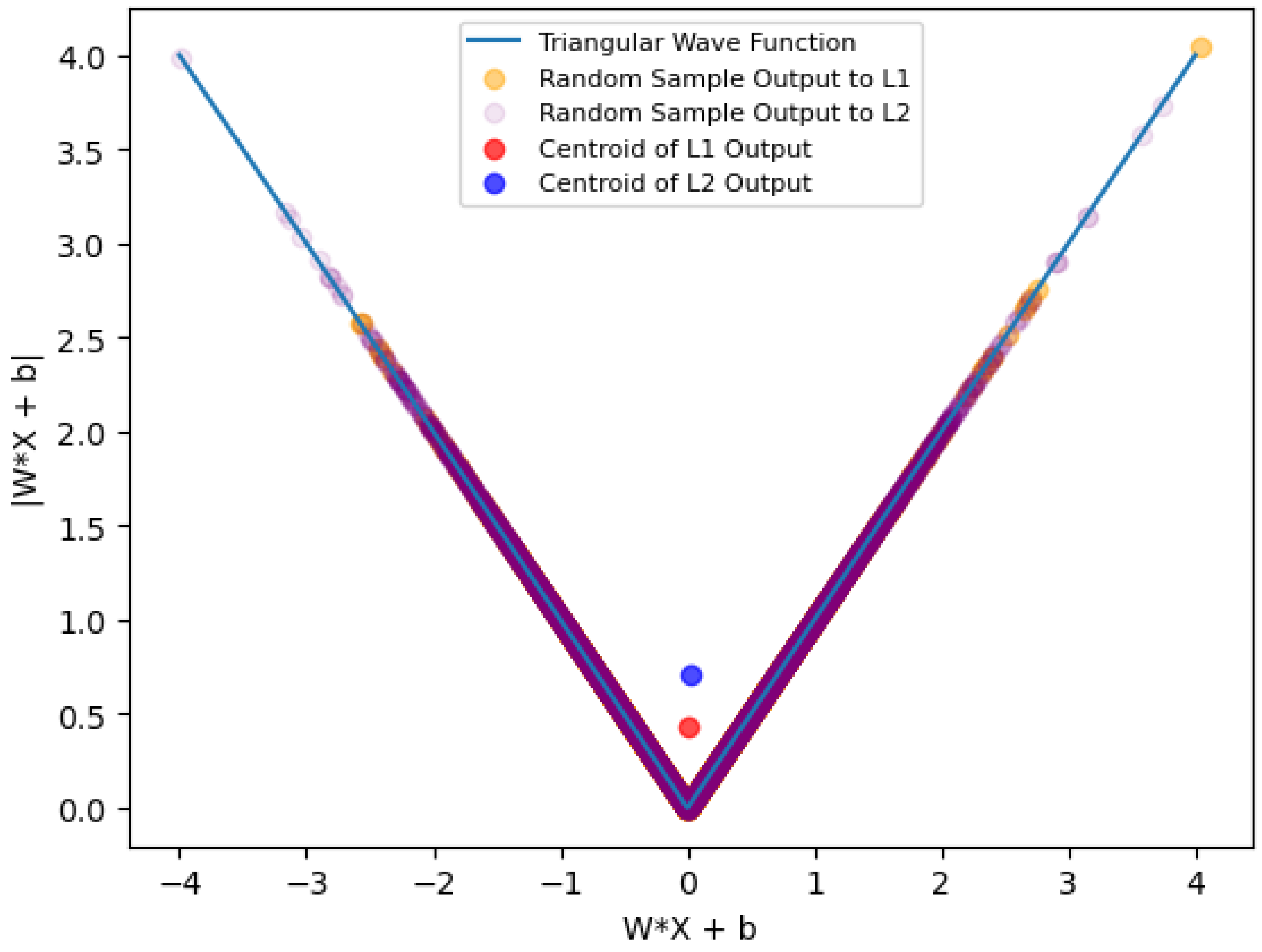


Figure 4: This graphic shows a sample of 10,000 points drawn from the test set and plots the distribution of the inner linear component operation (the input to the *Absolute Value* activation function) against the output of the activation function on the same point.

This second figure is included to show that this domain constraining artifact is not one that arises from the local structure of the Triangular Wave (which is an Absolute Value function on $[0, 1]$), but rather arises from the sinusoidal period.

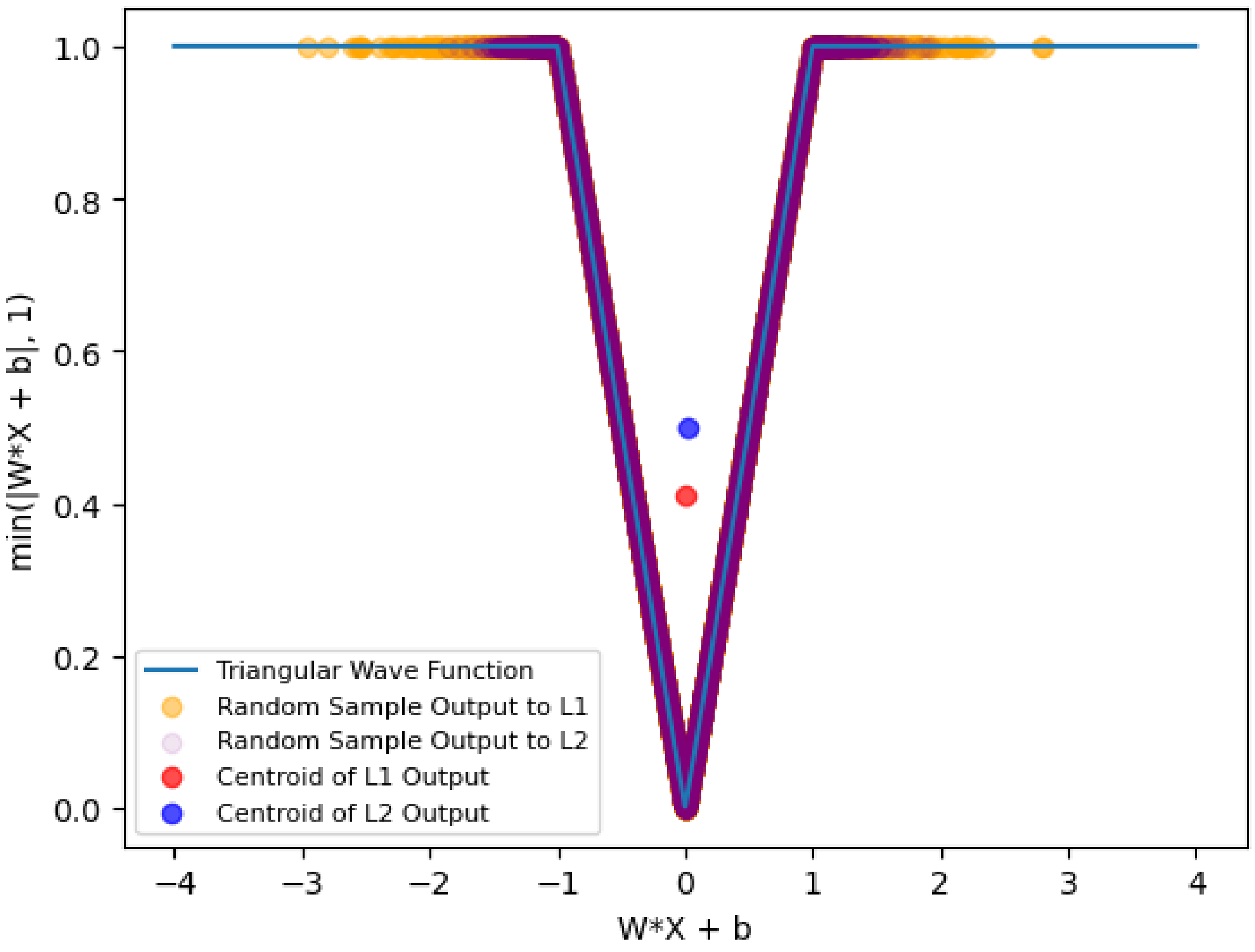


Figure 5: This graphic shows a sample of 10,000 points drawn from the test set and plots the distribution of the inner linear component operation (the input to the *Bounded Absolute Value* activation function) against the output of the activation function on the same point.

This third figure is included to show that this domain constraining artifact seems to instead arise from the fact that the *Triangular Wave* is bounded between $[0, 1]$.

5 Conclusion

In the layered "performance comparison by activation function" graphs it is clear to see that between the sinusoidal activation functions and the baseline ReLU there is a consistent difference; namely that very few of the loss curves for both train and test (actually only one) seem to get "stuck" at a certain loss and never decrease as training goes on. In all of the sinusoidal trajectories, there are a few hyperparameter combinations that never trend towards convergence and continue to exist around a consistent loss value for all epochs.

In spite of this, the best performing final accuracy comes from the Triangular Wave activation function which has a marginally better performance than all the other functions, of which Cosine consistently performs worse than all the others both in terms of average final metrics and standard deviation of average final metrics.

Furthermore, some insight is provided as to how these sinusoidal activation functions actually work and whether or not they utilize their full domain. It may be worthwhile to attempt a linearly increasing absolute value *Triangular Wave* function to see if changing the bounding property to one that provides a different output is useful to the network's learning capability (as is potentially already demonstrated in 4).

All of this is to say that further investigation is needed into this topic, but as shown by the results of the Triangular Wave activation function, there could be some merit in trying to optimize machine learning models with sinusoidal activation functions as it can potentially have more stable gradients.

References

[1] Wikipedia contributors, “Multilayer perceptron — Wikipedia, the free encyclopedia,” 2023, [Online; accessed 17-May-2023]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Multilayer_perceptron&oldid=1153920594

[2] ——, “Activation function — Wikipedia, the free encyclopedia,” 2023, [Online; accessed 17-May-2023]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Activation_function&oldid=1151093352

[3] A. Silvescu, “Fourier neural networks,” *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, vol. 1, pp. 488–491 vol.1, 1999.

[4] ——, “Fourier neural networks,” in *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, vol. 1, 1999, pp. 488–491 vol.1.

[5] A. Baldominos, Y. Sáez, and P. Isasi, “A survey of handwritten character recognition with mnist and emnist,” *Applied Sciences*, vol. 2019, p. 3169, 08 2019.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] L. Chen and J. Bruna, “On gradient descent convergence beyond the edge of stability,” 2022.