Exploring Generalized Trainable Activation Functions in Neural Networks

Aaron Davis
M.S. in Data Science
University of Colorado at Boulder
Boulder, Colorado
aaron.davis@colorado.edu

ABSTRACT

Activation functions are used to introduce non-linearity into neural networks to allow a network to learn complex non-linear functions by combining a series of simple non-linear transformations of linear functions with trained weights. Common activation functions include rectified linear units (ReLU and variants), sigmoid, and tanh functions. These activation functions are not learned but are chosen by the researcher based on experience and domain knowledge.

Creating trainable activation functions that replace researcher intuition and experience with trainable parameters is a field that has been around since the early 1990s [1], and still shows a great deal of potential for advancement. Our results show that generalized activation functions help increase training convergence speed while also allowing smaller networks to achieve substantially higher accuracy relative to networks with the same architecture that use fixed activation functions.

Keywords—activation functions, neural networks, machine learning

I. INTRODUCTION

The most popular activation functions currently used to introduce non-linearity into the hidden layers of neural networks are ReLU and Sigmoid. The functional form of the Sigmoid activation function is as follows:

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

In the case of a single output network with no hidden layers, "z" is a linear combination (with bias term) of trainable weights and training data covariates. The ReLU activation has the following functional form:

$$ReLU(z) = max\ (0, z)$$

Activation functions allow neural networks to learn complex non-linear functions, but each activation comes with some downsides. The sigmoid function and other saturating functions suffer from the vanishing gradient problem which can make network training infeasibly slow. The rectified linear unit, on the other hand, is incapable of distinguishing varying magnitudes of negative values. The goal of this paper is to explore removing these downsides by not imposing a restrictive functional form on neuron activations.

II. RELATED WORK

The entire goal of neural networks is to minimize some loss or maximize some reward. What if we connect the choice of activation function for each neuron directly back to this goal by including trainable parameters to modify the form of the function, not just the input? This line of thinking has yielded a series of interesting results that are summarized well in "A Survey on Modern Trainable Activation Functions" by Apicella, et al [2].

The results that come from thinking about activation functions like this includes a generalized form of the sigmoid function from Hu, et al. [1],

$$AGSig(z) = \frac{\alpha}{1+e^{-\beta z}}$$

where the parameter α scales the function output, and the parameter β scales the function input before transformation. Other common-sense trainable transformations to activation functions are exemplified by Qiu, et al. [3] in "Frelu: Flexible rectified linear units for improving convolutional neural networks ". The FReLU activation function is defined as follows:

$$FReLU(z) = ReLU(z+\delta) + \eta$$

where the trainable parameter δ controls a right or left shift of the ReLU function, and η controls a shift up or down.

All of these transformations add additional flexibility to what kind of transformation an activation function will make, but the result still assumes an overall activation form. AGSig still assumes an underlying sigmoidal form, and FReLU still assumes a rectified linear form. The premise of our research is that removing this assumption of a particular functional form will allow our network to learn more efficiently.

How can we remove this assumption, though? We want to design an activation that can learn any arbitrary transformation of the inputs, so that we can reduce our model's reliance on

human imposed activation form. This is a perfect use-case for neural networks. The idea here is to replace every neuron activation in a network with a one input/one output neural network that can replace the fixed activation function with one that trains with the rest of the network.

Figure 1 shows a single output neural network with no hidden layers and a fixed activation function "A".

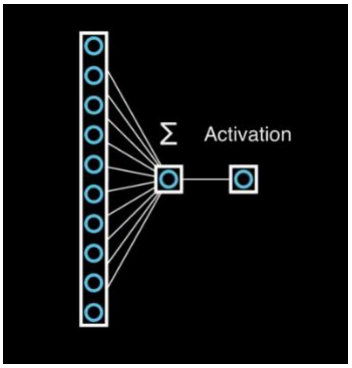

Figure 1: Traditional Fixed Activation Function

Figure 2 shows the counterpart network to the one shown in Figure 1 that replaces the single, fixed activation function with a single input/single output neural network of arbitrary size and architecture, that uses a fixed activation for all neurons in the sub-network.

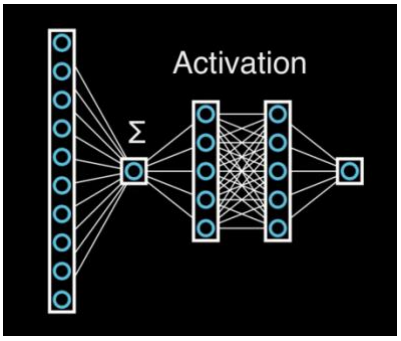

Figure 2: Arbitrary Trainable Activation Function

A fitting name for this type of arbitrary learned activation function that results from a composition of fixed activations and learned parameters is "meta-neuron", meaning "a neuron composed of other neurons".

After experimenting with these meta-neurons for several months, we discovered that something very similar had already been done in 2019 by Apicella, et al. [4], and they called this architecture "Variable Activation Function Subnetwork" or "VAF", for short.

Our architecture differs slightly from theirs in a couple of ways. First, they restrict a VAF to having one hidden layer. No such restriction exists for meta-neurons, and this introduces an interesting set of challenges that come with training very deep neural networks. This forces the introduction of residual connections between all adjacent hidden layers inside each meta-neuron to increase accuracy and decrease epochs to training convergence.

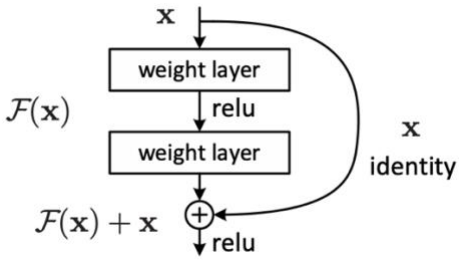

Figure 3: Residual Connection (From Original ResNet Paper) [5]

The second notable difference is that the single input and single output neurons for the VAF have no non-linear activation. Meta-neurons have activations for those neurons, as shown in Figure 2.

III. Testing and Results

We tested meta-neurons in multiple different model architectures. We also tested using standard feed-forward networks (modified to replace fixed activations with meta-neurons, of course) on the MNIST dataset [6]. We then compared the results with analogous neural networks with fixed activation functions.

3.1 Comparing Trained Activations and Fixed Activations

First, we compare the architectures shown in figure 1 and figure 2 on the MNIST dataset (except with 784 input features instead of 10, and with 10 output neurons instead of one). The sum and activation portion in figure 2 are analogous to the sum and activation portion in figure 1, except the architecture in figure 2 allows the activation to be an arbitrary function, rather than limiting it to the typical structure. The results of this comparison are as shown in Figure 4, subplot 1.

In Figure 4, the activation ("sigmoid" in the first subplot, for example) refers to the fixed activation used both in the figure 1 activation and the fixed activation used throughout the flexible activation subnetwork in figure 2.

Additionally, the neuron count refers to the number of neurons per hidden layer in the traditional network, and the number of meta-neurons per hidden layer in the network with trainable activations.

Finally, the number of layers again refers to the number of neuron layers in the traditional feedforward network and the number of meta-neuron layers in the network with trainable activations.

In Figure 4, all blue lines represent the accuracy over time for our meta-neuron networks, while orange lines represent the accuracy over time for a traditional feedforward network.

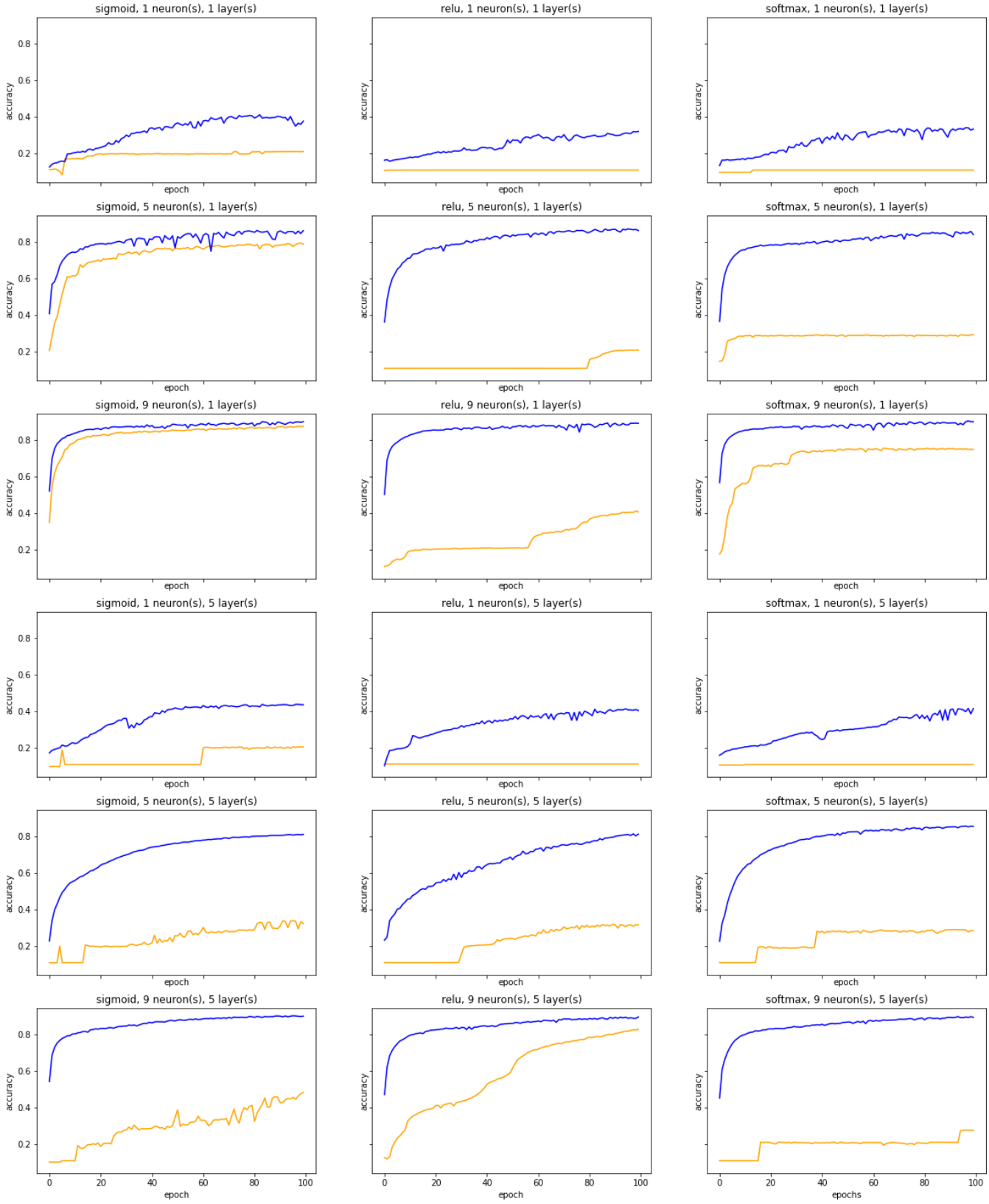

Figure 4: Comparing Fixed and Trainable Activation Functions
(Blue: Trainable Activation, Yellow: Fixed Activation)

Figure 4 clearly shows that regardless of model architecture tested and regardless of activation function tested, models with trainable activation functions always converged to higher accuracies significantly faster than models without trainable activation functions. Each model was trained for only 100 epochs, because accuracy is not the only metric we care about.

If trainable activation functions achieved overall better accuracy but required orders of magnitude more training, then the result would arguably not worth the additional time needed for training. Figure 4 shows that the trainable activations actually help increase speed of convergence and model accuracy, so this is not an issue here.

3.2 Comparing Meta-Neuron Models Against Each Other

But does adding extra layers or increasing layer width inside of the trainable sub-networks help, hurt, or have no effect on model accuracy and convergence speed? Figure 5 attempts to answer part of this question by fixing all aspects of the model except for the number of meta-neurons in each layer (or the width of the model, in other words). We then plot the training history for each of these models.

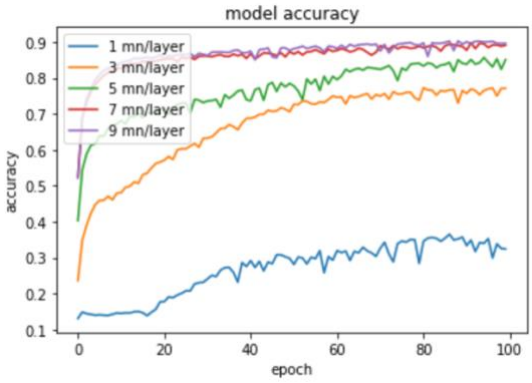

Figure 5: Comparing Number of Meta-Neurons per Layer

Figure 5 seems to show that increasing model width does help increase accuracy, convergence speed, and training stability, though for this dataset and this model architecture, the benefit of adding meta-neurons in a layer seems to become minimal around 7 meta-neurons per layer.

Does adding extra layers within each meta-neuron subnetwork help, hurt, or have no effect on model training? Figure 6 seems to show that for this dataset and model architecture, holding everything else constant and adding additional layers in each subnetwork has either no effect or a negative effect. This discovery seems to validate Apicella's [4] decision to limit VAFs to a single layer within the subnetwork. Adding additional layers within the subnetwork does not appear to have a positive impact and doing so may unnecessarily increase needed computing power.

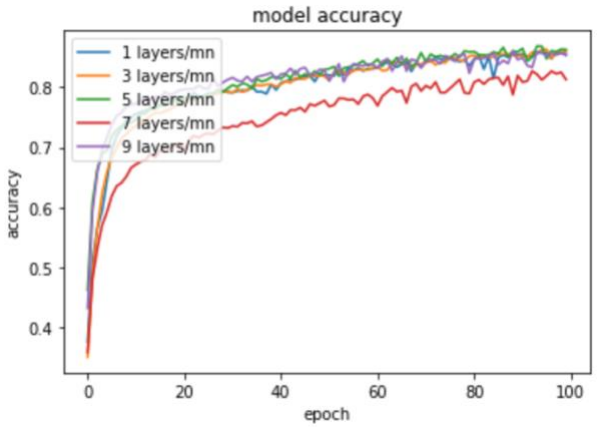

Figure 6: Comparing Number of Hidden Layers in Each Subnetwork

How does changing the number of neurons in each meta-neuron hidden layer change the accuracy and convergence speed of the model?

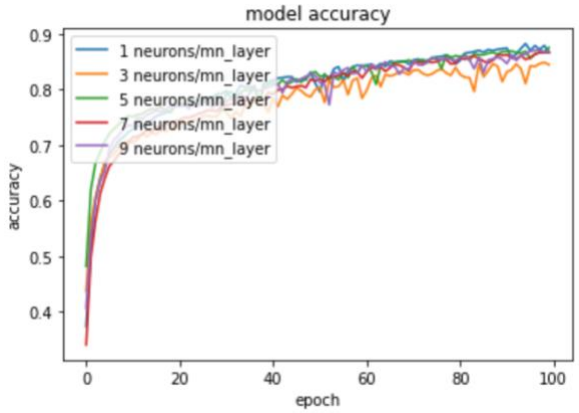

Figure 7: Comparing Number of Neurons per Subnetwork Hidden Layer

Figure 7 appears to show that for this architecture trained on the MNIST data over 100 epochs, the results are inconclusive. These results could be more pronounced on a different dataset or with a different model architecture.

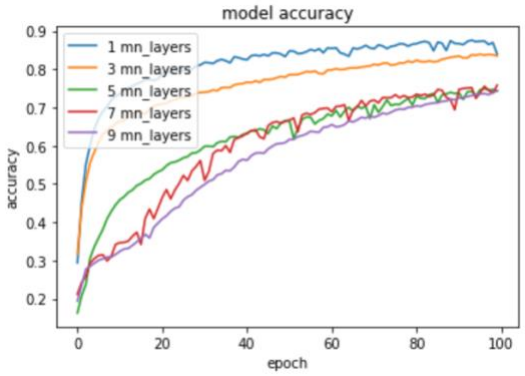

Figure 8: Comparing Number of Meta-Neuron Layers in Network

Figure 8 shows that adding increasing meta-neuron model depth has a negative effect on convergence speed and accuracy. This is a well-known issue with very deep neural networks that don't use residual connections. Here, we use residual connections within the layers in each meta-neuron subnetwork, but do not use residual connections across meta-neuron layers. As a result, it is unsurprising that increasing the number of meta-neuron layers without residual connections across meta-neuron layers in the model would decrease accuracy and convergence speed.

The next figure (Figure 9) appears to show that on this data and with this model architecture, ReLU activation functions used for the fixed activations with each subnetwork cause the network to converge more slowly than all other activations listed. ReLU normally performs very well because it is non-saturating, so we believe this may not be representative of general performance when used in meta-neurons.

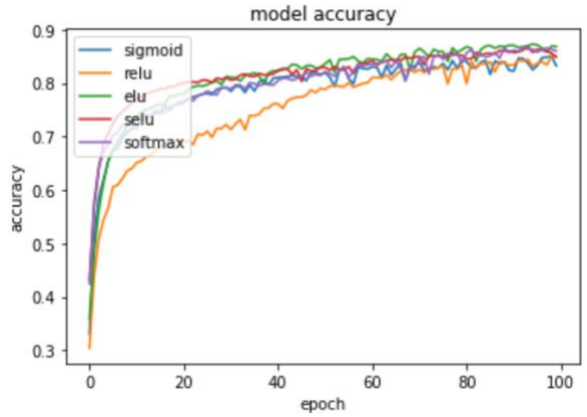

Figure 9: Comparing Fixed Activation Functions Used in Each Subnetwork

The next figure (Figure 9) appears to show that on this data and with this model architecture, ReLU activation functions used for the fixed activations with each subnetwork cause the network to converge more slowly than all other activations listed.

3.3 Visualizing Learned Activation Functions

Each single-output/single-input subnetwork learned some activation function for each of these examples. In Figure 10, we show the activation learned in a network with a single meta-neuron layer (with one hidden layer with 5 neurons in the hidden layer, and activated throughout with ReLU), with 3 meta-neurons in the single layer.

The results in Figure 10 are rather surprising in that only the first learned activation is obviously non-linear. The network appears to have learned linear activations for the second and third network.

This seems odd since the flexibility of neural networks is found in their non-linear activations, but if a network learned linear activations for certain networks, that may be strong evidence that those neurons can be pruned from the network and replaced with simple constant multiplications, rather than activation. This may offer potential reductions in required computing power at inference time.

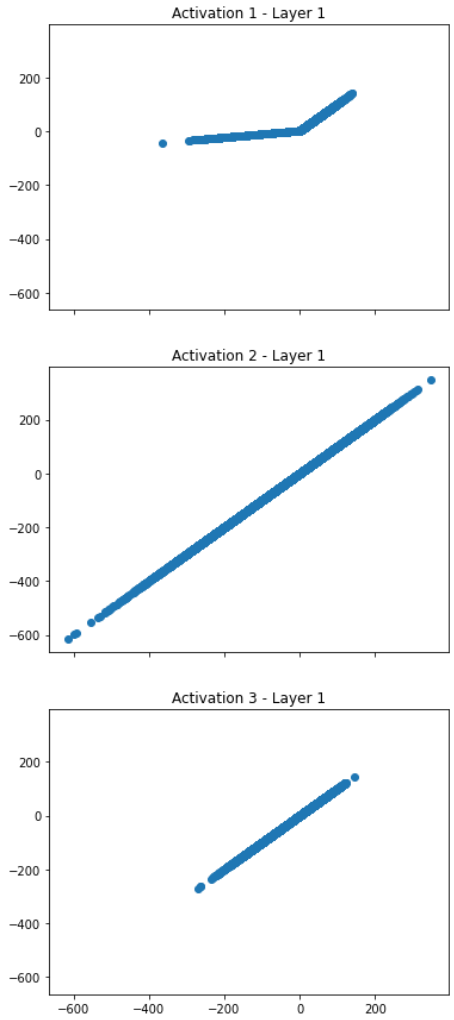

Figure 10: Visualizing Trained Activations

IV. Future Work and Conclusion

4.1 Future Work and Research Opportunities

Further exploration is required on converting learned activations to fixed activation post-training to decrease computing power required to run inference. This work could potentially done by creating spline approximations of the learned activation, in the case of smooth learned activations.

Additionally, future work includes comparing meta-neurons to VAFs, FReLU, AGSig, and other related learned activation functions. Using meta-neurons for learned activation functions may theoretically provide incredible flexibility, in practice it may cause increased computing requirements without accompanying increases to accuracy over other learned activations.

There seems to be potential to combine the flexibility of FReLU with the flexibility of AGSig, by adding parameters to 1) stretch or squish the x-axis, 2) stretch or squish the y-axis, 3) move the activation right or left, and 4) move the activation up or down. AGSig and FReLU each contain a couple of these abilities, but neither have all four so this is an area that has potential for significant future work.

4.2 Conclusion

Flexible activation functions show significant promise in improving model accuracy and convergence speed over a fixed number of epochs. Meta-neurons clearly show a significantly improved convergence speed and higher accuracy relative to traditional neurons in feed-forward networks on the MNIST dataset.

Meta-neurons are a generalized form of VAFs, and after a good deal of testing they did not seem to provide any significant increase in accuracy or convergence speed over the simpler VAF model proposed by Apicella, et al. [4] on this dataset. Further testing is required to discover if this is generally true.

REFERENCES

[1] Hu, Z. "The study of neural network adaptive control systems." Control and Decision 7 (1992): 361-366

[2] Apicella, Andrea, et al. "A survey on modern trainable activation functions." Neural Networks 138 (2021): 14-32.

[3] Qiu, Suo, Xiangmin Xu, and Bolun Cai. "FReLU: flexible rectified linear units for improving convolutional neural networks." 2018 24th international conference on pattern recognition (icpr). IEEE, 2018

[4] Apicella, Andrea, Francesco Isgrò, and Roberto Prevete. "A simple and efficient architecture for trainable activation functions." Neurocomputing 370 (2019): 1-15.

[5] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6] LaCun, Yann, et al. “The Mnist Database.” *MNIST Handwritten Digit Database*, http://yann.lecun.com/exdb/mnist/.