Comparison of Different Activation Functions in Neural Network Training for Cyber-Physical System Applications

Jefferson Patrick Woolard

***Abstract*— As artificial intelligence technologies such neural networks become more common place in general purpose software development, important considerations need to be made as to what constitutes as the highest performing parameters and features when implementing neural network solutions to problems in software. This paper considers one such feature, the activation functions present in neural network models and architectures, and contrasts their performance with respect to classification problems necessary for cyber-physical system applications. Metrics such as loss functions will be used to determine how well known activation functions such as sigmoid, ReLU, hyperbolic tangent, and more affect neural network training performance for cyber-physical system applications. Results, analysis, and justifications thereof for an activation function performance comparison experiment are provided.**

***Keywords*— Feed-forward neural network, vision-based navigation, activation functions, cyber-physical systems, neural network classification problems, loss functions**

I. Introduction

With the emergence of machine learning technologies becoming increasingly more prevalent and advanced, the field of software development and engineering now more than ever has access to more ways to solve modern problems of high complexity using new artificial intelligence techniques. This especially true in the world of cyber-physical systems where the performance of machine learning tools such as feed-forward neural networks have been demonstrated on numerous occasion to outperform the capabilities of humans. Notable examples of feed-forward neural networks being utilized in cyber-physical applications that involve localization, classification, and fusion of data tasks of autonomous vehicles such as self driving automobiles [1] and aircraft collision avoidance systems [2].

In particular for the usage of feed-forward neural networks, there is an urgency to implement any such machine learning applications with emphasis on ensuring that adjustable parameters of the neural network are selected to be the most optimal with respect to the given task at hand. One such adjustable parameter of a feed-forward neural network's architecture is the activation function assigned to neurons in the model. Informally, activation functions in neural networks (also referred to as transfer functions or nonlinearities) can be described as what is used to transform a set of inputs (or some calculation over some set of inputs) into outputs that are then fed to another layer of neurons in the network [3]. There exist rich literature [4] [5] confirming the notion that the particular selection of activation function can significantly impact the overall performance of a neural network in regards to accuracy (such as the ability to correctly identify input in the task of classification problems).

This paper will examine a practical comparison of well known neural network activation functions for the use of constructing a well trained model for a particular cyber-physical system application scenario. The scenario in question will be training a model for a classification type problem of recognizing hand gestures from a flight drone mounted camera to inform navigation tasks. There exists a public GitHub repository [6] hosting the code for the particular cyber-physical system application as well as the code used for training the model utilized in the scenario. The machine learning aspects of the application will be implemented in python using Tensorflow's ML api for python [7] and the Mediapipe ML platform api for fast hand keypoints recognition [8].

The comparison results made between the activation functions will be tabulated numerically using the metric provided by loss functions which also have an existing rich literature [9] [10] as to why they are a preferred metric to use to describe the performance of a trained model. Efforts will be made to define in more formal and mathematical detail the activation functions selected, the precise neural network model, and the loss function used for experimentation. One of the main objectives of this paper will be analyzing the results obtained, postulate as to why the performance of the activation functions resulted thusly, and what said results could mean for other classification task oriented cyber-physical system applications and scenarios that use feed-forward neural networks where considerations on activation functions need to be made.

II. Feed-Forward Neural Networks and Activation Functions

In order to examine further the importance of the role of activation functions in the complex structures (i.e., layers of interconnected groups of computational nodes inspired by biological neurons) of feed-forward neural networks (FFNNs), a generalized formal mathematical definition [11] is provided:

$$\mathbf{y_i} = f(\sum\nolimits_{\mathbf{j=1}}^{\mathbf{n}} \boldsymbol{\omega}_{ij} x_j + \theta_i) \tag{1}$$

The output (also called behavior) of a neuron is inherently described by the activation function $f(\cdot)$ in the form of the

Author is with the School of Computer and Cyber Sciences, Augusta University, Augusta GA 30912 USA. Email: jwoolard@augusta.edu

equation above where x_j is understood as jth input of the ith neuron, ω_{ij} is understood as the weight from the jth input to the ith neuron, θ_i is understood as the bias of the ith neuron, and $\mathbf{y_i}$ is understood as the output of the ith neuron which is directly determined by a given activation function [11].

For the particular classification task of recognizing hand gestures to inform navigation controls of an autonomous flight drone, we will consider the model diagram [6] provided in Fig. 1. The FFNN model used in the application will take a take a one dimensional list of forty-two x and y coordinates values (which have been normalized) relating to keypoints located on a hand in the camera view of the autonomous flight drone which gets propagated throughout the model before the Softmax function selects the gesture class with highest probability of matching our selected gesture. A given activation function $f(\cdot)$ will determine the output of each neuron located within a hidden layer. We will substitute $f(\cdot)$ with the activation functions we wish to compare and see to what degree this affects the performance of the FFNN. This performance metric will primarily be measured in loss which will be more thoroughly defined later in this section. To maintain consistency between each test of activation function performance for comparison, the size and structure of the model with respect to number of neurons and the amount of hidden layers as well as the data used for model training will remain the same as to isolate only the activation function as the determining feature of performance.

Of course, the motivation of this paper being to determine which such activation functions perform the best for FFNN models utilized in the hand gesture recognition task (and the implication of the results thereof to similar cyber-physical system scenarios), we will examine formal mathematical definitions and justification for selection of the activation functions used in our experimentation as follows.

The Sigmoid activation function generates values from 0 to 1, and is one of the most widely known and often used activation functions used in neural network applications. This is due to its many mathematical properties such as nonlinearity, computational simplicity of its derivative, and influence of speedy backpropagation [12]. Its formal mathematical definition [5] is as follows:

$$\mathbf{y} = \frac{1}{1 + e^{-x}} \tag{2}$$

The ReLU activation function generates 0 for any non positive input values and the input itself otherwise. In recent history, ReLU has become a widely preferred activation function, especially for the use in deep-learning applications due to its simplicity and effectiveness. It has been observed in real-world applications that the use of the ReLU activation function can actually provide a significant speed up in training compared to sigmoid [5]. However, despite the observed speed performance, ReLU can suffer from an issue known as the Dying ReLU problem where if too many inputs to a neuron (where the ReLU activation function is being applied) in the network are negative, then

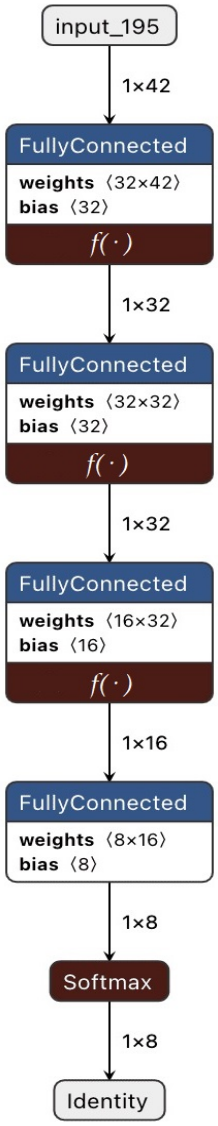


Fig. 1. Exact FFNN Model used for Hand Gesture Recognition Application

learning performance of the network can suffer due to issues that can arise during backpropagation [13] [14].

ReLU's formal mathematical definition [5] is as follows:

$$\mathbf{y} = max(0, x) \tag{3}$$

The Swish activation function is mathematical defined [15] as follows:

$$\mathbf{y} = \frac{x}{1 + e^{-x}} \tag{4}$$

As is apparent, this activation function is obtained by multiplying the input and sigmoid function together. This provides swish with preferable gradient properties that allow for quick training similar to ReLU, but unlike ReLU, swish

is a non-monotonous function. This property of being non-monotonic may or may not provide benefits to loss performance, so it has been included in the comparison experiment in the following section to investigate any such potential benefits.

The Hyperbolic Tangent (tanh) activation function generates outputs from -1 to 1, providing an anti-symmetric property relative to the origin that may provide loss performance benefits worth investigating in the experimental section. Its formal mathematical definition [5] is as follows:

$$\mathbf{y} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

Lastly, the exponential activation function is defined simply as $y = exp(x)$ or simply e^{x}. This function was also selected for experimental comparison as not only is it computationally simple to produce, but its properties of being entirely nonlinear, monotonically increasing, and unlimited from the top graphically, make it unique in some way to every other activation function we wish to compare. This unique combination of properties about the exponential activation function may or may not provide loss performance benefits that are investigated in the experimental section following.

Now that justification has been provided as to why the activation functions being compared were selected, we can more thoroughly evaluate our loss performance metric and the importance thereof. The objective of loss minimization is one of the most crucial aspects of training neural networks, as loss is a metric that describes numerically how far off a model's output is from what the actual true output of the model should be. In general, the lower the computed loss, the better the model is at performing its intended task [16]. Functions that compute loss can be utilized in adjusting model weights for neural network training, and are also used to evaluate the performance of a model when processing testing data, thus making them an ideal candidate for comparing model performance with respect to varying activation functions. For the application of identifying and classifying hand gestures to inform drone navigation directions, the Sparse Categorical Cross-Entropy (SCCE) loss function will be used in the experimental comparison as it is designed specifically for classification problems and has already been implemented in the Tensorflow api used for training the model [17]. A formal mathematical definition [16] of how SCCE computes loss has been provided,

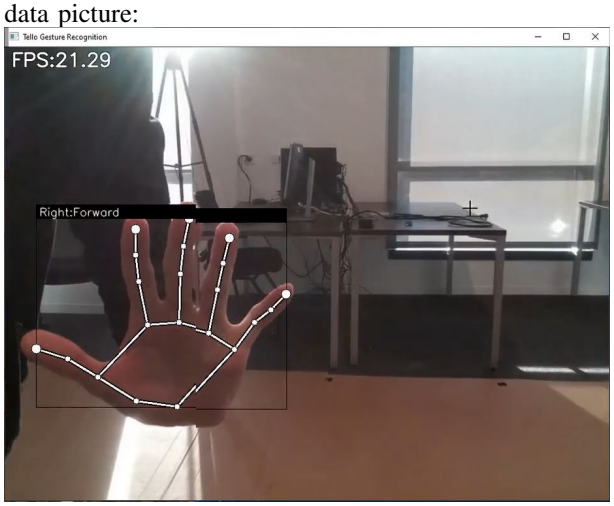
$$L_{SCCE} = -\sum_{\mathbf{i=1}}^{\mathbf{n}} t_i \log(p_i), \tag{6}$$

where n is the number of classes (i.e. hand gestures) we wish to be able to identify, t_i is a truth label corresponding to the actual gesture, and p_i is the Softmax probability generated as output from the model.

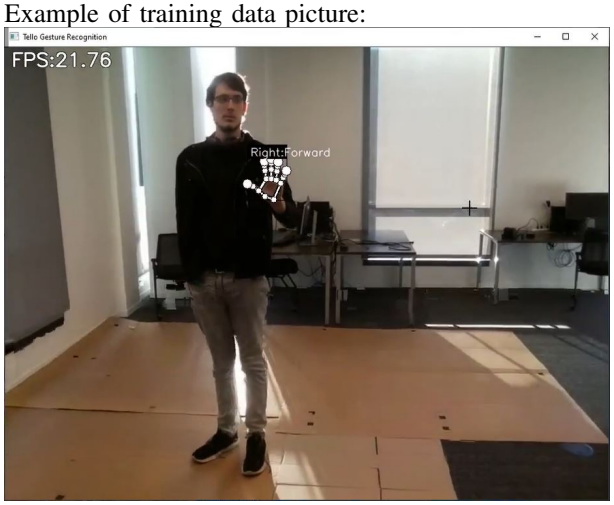
III. Experimental Set-up and Results

For the comparison results, two different performance experiment types will be used to conduct tests for each of the activation functions mentioned in the prior section. Each experiment type will use the same size and structure of neural network model, the same training data, and the same number of 1000 maximum training epochs (training may terminate early if after 50 consecutive epochs loss does not improve). Again, this is to ensure that the only performance determining feature across each model is the activation function (denoted as $f(\cdot)$). The two experiment type tests are as follows:

- **Normal**, where the training data will be keypoint coordinates taken from pictures of a normal hand performing various hand gestures. Example of training data picture:



- **Distance**, where the training data will be keypoint coordinates taken from pictures of a hand more distant in the image performing various hand gestures. Example of training data picture:



These two experiment types are meant to demonstrate different realistic situations that may arise in cyber-physical system applications and to further demonstrate potential difference in the loss performance metric for activation functions across varying scenarios. For both experiment types, for each activation function, the performance test is run

$f(\cdot)$	Best L_{SCCE}	Average L_{SCCE}	Exp. Type
Sigmoid	7.41E-04	3.41E-03	Normal
ReLU	7.40E-04	1.17E-03	Normal
Swish	**2.35E-04**	1.28E-03	Normal
Tanh	3.73E-04	**1.06E-03**	Normal
Exponential	3.45E-04	0.148E-00	Normal
Sigmoid	**6.27E-07**	4.98E-04	Distance
ReLU	2.02E-04	6.68E-04	Distance
Swish	3.47E-04	**4.62E-04**	Distance
Tanh	5.14E-04	9.06E-04	Distance
Exponential	5.56E-03	7.23E-02	Distance

Fig. 2. Performance Results of Activation Function Experiments

three separate times. Both the best and average loss values (denoted with L_{SCCE} accordingly) have been recorded in Fig. 2.

To ensure integrity of results, all training data that was used for experimentation and results is provided in a public GitHub repository [18] so that any replication of the experiment can be performed (To run experiment, use model training guide provided in hand gesture GitHub repository [6]). Special attention was also made for the collection and use of data in the experiment to avoid any statistical bias being present in the testing and results thereof. One such example of this intended aversion to statistical bias present in the experiment is the 1000 epoch upper bound on training (as well as halting training early should loss not improve after 50 consecutive epochs). These measures are in place to avoid any potential overfitting from over training the neural network model, which might otherwise influence the measurement of loss performance during the evaluation and model testing portion of the experiment [19].

By examining Fig. 2. some observations can be made concerning the comparison results (lowest loss values per column and experiment type are in bold). The range of loss values across the columns are (generally) within the same relative numeric range, showing that for models of this size and architecture, selection of these activation functions will produce a smaller influence on loss performance as opposed to a fundamentally dramatic one. Holistically, we can observe that swish activation function seemed to perform the best overall by having the best average performance in the distance training data experiment type, and by performing very closely to the best average performance (and having the best individual test performance) in the normal experiment type. Speculatively, it could be that the preferable gradient properties of swish (like ReLU which also performed considerably well) offer higher performance in loss when compared to the other functions. The impressive performance of swish in these experiments would agree with existing research done by Google and others that suggests its mathematical formulation alleviates the vanishing gradient problem present in ReLU with the Dying ReLU problem [15] [20].

More objectively, the exponential activation function performed poorly in both experiment types, showing that its unique set of properties offered no advantages in generating better loss performance. The poor performance of the exponential activation function can likely be attributed to its instability and the explosive gradient that occurs from consequent layers with exponential activation. Summarily, the main ideas that can be observed from the results are that activation functions with stable gradient properties (such as swish) are preferable and perform, on average, better with respect to loss than activation functions with unstable gradient properties (such as exponential).

IV. Conclusion and Future Directions

This paper explores loss performance comparisons for various activation functions for neural network applications used in cyber-physical systems. A clear testing and evaluation methodology has been provided, including justifications for the experimental comparison tests and for the activation functions selected. Comparison results were produced for a real world hand gesture flight drone navigation application and inferences with justification of said results have also been provided. Observations of comparison results concluded that average loss performance is influenced heavily by the gradient properties of the activation function used.

Future areas of research could include how activation function comparisons are affected by factors such as vanishing or exploding gradients across models of varying size complexity. Future considerations could also include how neural networks perform with more varied scenarios for training and testing cyber-physical systems. For example, in the application used for this paper, scenarios such as gloved hands, dynamic lighting, and varying skin tones could be investigated to determine if they impact model behavior.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[2] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10, 2016.

[3] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 04, pp. 310–316, 05 2020.

[4] A. A. Alkhouly, A. Mohammed, and H. A. Hefny, "Improving the performance of deep neural networks using two proposed activation functions," *IEEE Access*, vol. 9, pp. 82249–82271, 2021.

[5] Y. Koçak and G. Üstündağ Şiray, "New activation functions for single layer feedforward neural network," *Expert Systems with Applications*, vol. 164, p. 113977, 2021.

[6] N. Kiselov, "Dji tello hand gesture control." https://github.com/kinivi/tello-gesture-control, 2021.

[7] "Api documentation : Tensorflow core v2.6.1." https://www.tensorflow.org/api_docs.

[8] "Mediapipe: Cross-platform and customizable ml solutions." https://google.github.io/mediapipe/.

[9] R. D. Reed and R. J. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1998.

[10] J. Brownlee, "Loss and loss functions for training deep learning neural networks," Oct 2019.
[11] H.-D. Tran, W. Xiang, and T. T. Johnson, "Verification approaches for learning-enabled autonomous cyber-physical systems," *IEEE Design Test*, pp. 1–1, 2020.
[12] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *From Natural to Artificial Neural Computation* (J. Mira and F. Sandoval, eds.), (Berlin, Heidelberg), pp. 195–201, Springer Berlin Heidelberg, 1995.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.
[14] K. Leung, "The dying relu problem, clearly explained," Sep 2021.
[15] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017.
[16] K. E. Koech, "Cross-entropy loss function," Nov 2021.
[17] "tf.keras.losses.sparsecategoricalcrossentropy : Tensorflow core v2.7.0."
[18] J. P. Woolard, "Activation function comparison training data." https://github.com/JwoolardAU/Activation-Function-Comparison-Training-Data/tree/main, Nov 2021.
[19] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, p. 022022, feb 2019.
[20] S. Serengil, "Swish as neural networks activation function," Feb 2020.