International Journal of
Mathematical Modelling & Computations
Vol. 15, No. 02, 2025, 01- 16
DOI: 10.71932/ijm.2025.1205868

Parametric Activation Functions for Improved Identity Verification in IoT Devices: A Deep Learning Approach

S. Sarabadan [a,*], M. Mousavi [a]

[a] *Department of Mathematics, Imam Hossein University, Tehran, Iran.*

Abstract. One of the most important solutions in designing the architecture of a deep neural network is to use suitable activation functions in the hidden layers of the network. This function plays an important role in the back propagation algorithm and the calculation of the gradient of the cost function is based on the output of the activation function. In this paper, we will model a deep neural network to address an application problem in the Internet of Things, using experimental data recorded in a smart home, with the goal of identifying and preventing unauthorized devices from entering the Internet of Things network. The method used in this study relies on the radio frequency fingerprint of a radio device connected to the Internet of Things. The database used in this study consists of 8000 samples from 15 test series, collected using the One RF Hack radio receiver in the smart home and on 4 different connected devices. Finally ,we evaluated the performance of different activation functions in the hidden layers of the network. Ultimately, the most effective activation function was selected for the efficient and effective network. The Python code of the network architecture is located in GitHub https://github.com/SaeidSarabadan/RF_with_-ANN.

Received: 05 May 2025; Revised: 30 June 2025; Accepted: 09 July 2025.

Keywords: Deep neural network; Machine learning; Activation functions; Network performance improvement; Internet of Things; Radio frequency fingerprint.

Index to information contained in this paper

1. Introduction

Deep learning is a subset of machine learning that involves the use of artificial neural networks to model and solve complex problems [5]. In a deep learning system, there are usually several layers of artificial neurons, each layer processing and transforming the input data representation. This allows the system to learn hierarchical representations of data, with early layers learning low-level features and later layers learning higher-level abstractions [1,6]. The word depth refers to the number of layers through which data is transformed during processes. Artificial neural network consists of input layer, hidden layer and output layer [24]. Activation functions in hidden layers are crucial for deep neural networks. They introduce non-linearity, improving model capacity and interpretability.
Without activation functions, neural networks would be simple linear models [2]. Non-linear activation functions help avoid vanishing gradients [10]. Common activation functions include Sigmoid, ReLU, Swish [9,26], Leaky ReLU, and Softmax. Choosing the right activation function depends on the problem, data, and model architecture. In recent years, many efforts have been made by deep neural network developers to design new

*Corresponding author. Email: s.sarabadan@yahoo.com

actuator functions to be a more effective and efficient alternative to traditional activation functions [10, 16,22].

The proliferation of mobile devices, Internet of Things devices and the increasing importance of security have led to more advanced and secure authentication methods [3]. One of these methods is using radio frequency fingerprint [18,19]. Radio frequency (RF) fingerprinting is a technique used to identify and classify radio frequency signals based on their unique characteristics, similar to human fingerprints. The goal is to extract a unique set of features from the RF signal that can be used to identify the source, type, or behavior of the signal. One of the applications of radio fingerprinting is the identification and classification of RF signals to identify and prevent unauthorized access to networks connected to the Internet of Things in a private area. The steps of using radio fingerprint are: signal capture, signal processing and finally extraction of unique features, which is a set of numerical values that represent the signal. The applications of deep learning in radio frequency are many and varied [3,23,35]. Artificial intelligence algorithms can be trained to classify RF signals based on their characteristics to enable identification of specific devices, modulation schemes, or transmission protocols.

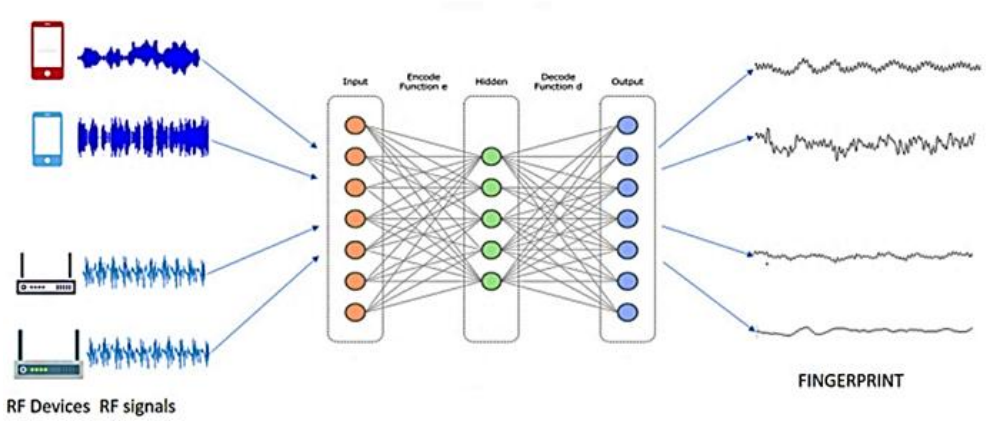


Figure 1. Deep Learning for RF Fingerprinting.

In this article, an attempt has been made to examine the properties, functions and applications of different types of activation functions in a neural network and to describe the latest research achievements in this field. Finally, we model a deep neural network on a practical problem with laboratory data recorded by a private company, whose purpose is to authenticate 4 IoT devices using radio fingerprints.

After designing the network architecture, using different activator functions with appropriate hyperparameters in the hidden layers of the network, network performance has been evaluated and the best activation function has been selected for effective and efficient network.

2. Methodology

2.1. *Deep learning and using empirical data*

The method of using data has undergone many changes after the spread of intelligent models among researchers and analysts [30]. Today, models based on classical mathematics have lost the ability to process and work with huge amounts of data [1,13,14]. The role of the emerging phenomenon of technology and the computer field, i.e. artificial intelligence, has emerged in data processing. Where the traditional analytical relationships

in estimating and processing data that require thousands of parameters in their equations will not be able to respond.

Deep learning is one of the algorithms in the field of machine learning and the newest achievement in the field of artificial intelligence, which has received attention due to its different applications in modeling science and technology issues, and has attracted different trends of science [6,24]. In deep learning, the architect of the neural network by deepening the network or in better words, adding hidden layers to the network enables it to learn more complex algorithms and receive a huge amount of data.

Each deep learning model generally consists of input layer, hidden layer and output layer. In each layer, depending on the type of network architecture, a number of neurons are placed. In the deep learning model, by going from each layer to another layer, the weighted sum of the set of neurons of the previous layer is calculated and transferred to the next layer. In this case, the outputs of the layers become linear and all layers become a similar layer. In this case, the neural network only models a linear function and will not be able to distinguish and learn nonlinear boundaries between classes.

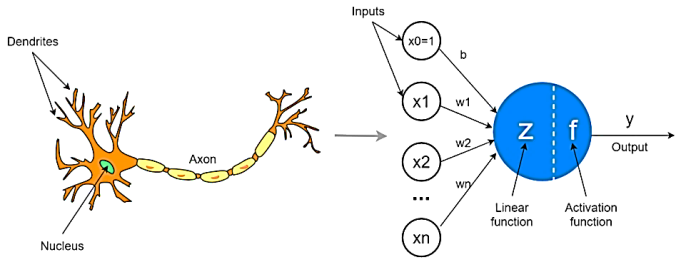


Figure2. Human neural network vs Deep Learning.

In deep neural network, it is necessary to use the activation function to activate the outputs of each layer. If the activation function is not used, the number of hidden layers will not affect the learning of the model. In fact, not using the activation functions only creates a linear equation of weights and biases which cannot help us to solve complex problems. A neural network without an activation function is just a linear model [6].

In fact, learning in deep neural network by going from each layer to another layer, the weighted sum of the set of neurons of the previous layer is calculated and transferred to the next layer by applying a non-linear activation function. But how each weight should change is a bit complicated. Because the error obtained in the output of the final layer can be caused by the neurons of the last layer as well as the neurons of the previous layers. These weight changes are done by the backpropagation algorithm [2]. The error contribution of each neuron in each specific layer and the process of updating the weights continues until the best coefficients are reached.

2.2. *Back propagation algorithm*

Backpropagation is an essential algorithm in machine learning and deep learning that is used to train artificial neural networks. It is a supervised learning method, which means that the model is trained on labeled data and the goal is to minimize the difference between the model predictions and the actual labels [7]. The back-propagation algorithm consists of two main steps:

1.Forward pass: In this step, the input data is propagated through the network to calculate the output predictions.

2.Backward pass: In this step, the error between the predictions and the actual labels is calculated and propagated in reverse in the network to update the model parameters.

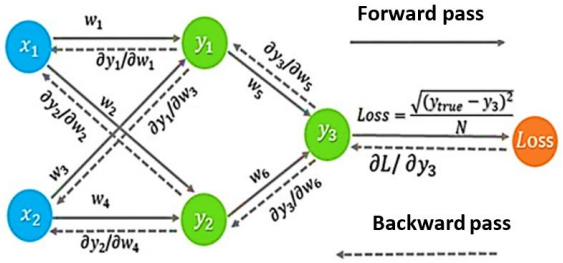


Figure3. Forward pass and Backward pass.

2.3. *The role of the derivative of the activation function in backpropagation*

In the backward pass, by calculating the gradient of the cost function according to each parameter, the error is propagated backward in the network. These gradients are used to update parameters using an optimization algorithm such as gradient descent. The gradients of the cost function determine the amount of changes according to parameters such as activation function, weights, bias and other relevant items [8]. The derivative of the activation function is used to calculate the gradient of the cost function according to the inputs of the activation function and plays an important role in the backward pass.

Not paying attention to the derivative of the activation function causes two basic problems in network learning, vanishing gradient and exploding gradient.

- **Vanishing gradient problem or neuronal death**

Some activation functions map their large input values to the range 0 to 1. Therefore, the output of the derivative value of the function becomes a very small number. If the activation function is used in several consecutive layers, this causes the gradient value to decrease exponentially and approach zero. The gradient value approaching zero causes the neural network parameters such as weights and bias values in the initial layers of the network to not be updated, and therefore the learning of the network is not done properly. In fact, the neuron loses its function and therefore this state is called neuronal death [6].

- **Exploding gradient problem**

The problem of gradient explosion occurs due to the exponential growth and repeated multiplication of large gradient values, in this case, the gradient values become more than 1. This issue makes the amount of updating weights very large, which will lead to network instability. Following the explosion of gradients, network learning stops and network weights are not updated.

3. Activation Function

Due to the central role of activation functions in deep neural networks, the introduction of new functions has been the interest of many researchers in recent years [5,2,28,34]. Many of trainable activation functions have recently been proposed and there has been a lot of interest in this topic for years, as illustrated by the chart below.

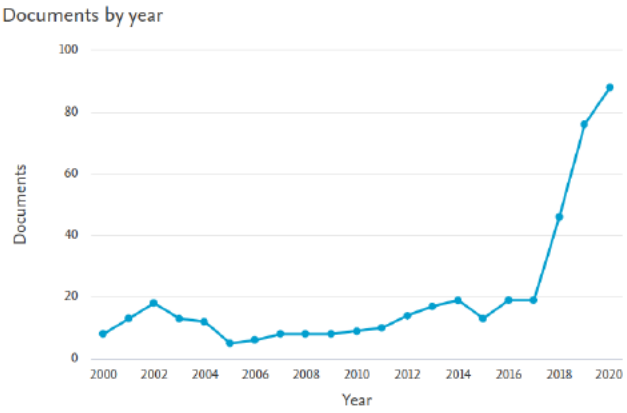


Figure 4. Number of papers by year on trainable activation functions.

Activation functions are actually like gates that exist in every neuron. The input of this gate is the input of each neuron in each layer and its output is transferred to the next layer. The activation function decides whether each neuron is activated or not and how its output values are expressed if activated. Neural networks use activation functions to help the network learn complex data and provide acceptable predictions in the output.

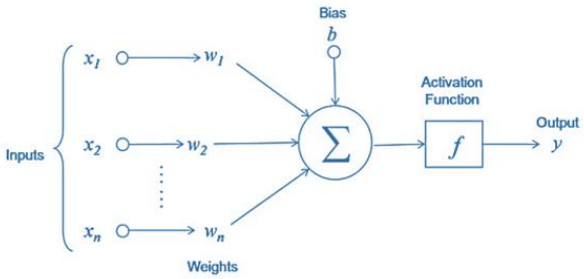


Figure 5. Activation functions like gates.

Non-linear activation functions allow the model to adapt to complex and non-linear data[10,21]. This is very important for complex data such as photos, videos, signals, etc. In this paper, we propose a possible taxonomy of activator functions. This classification is based on the possibility of changing the shape of the activation function during the training phase.

We can separate two main categories:

3.1. *Fixed-shape activation functions*

Fixed parameter activation functions, or in other words, nonparametric activation functions, are fixed functions that do not rely on any parameters that need to be estimated or learned from data. All activation functions with a fixed form, for example, all classical activation functions used in neural networks, such as Sigmoid, SoftMax, ReLU, fall into this category [12,27,28].

- **Sigmoid activation function**

Sigmoid function is a real, bounded and differentiable function that can be defined for all real values and has a positive derivative [29]. Show the formula of this function as follows:

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

The domain of the sigmoid function includes all real numbers and the range of this function changes uniformly from 0 to 1 or from 1 to -1 depending on the type of function. The derivative of the sigmoid activation function is as follows:

$$S'(x) = sigmoid(x) \times (1 - sigmoid(x))$$

The sigmoid function is considered a monotonic function, but the derivative of this function is not a monotonic function.

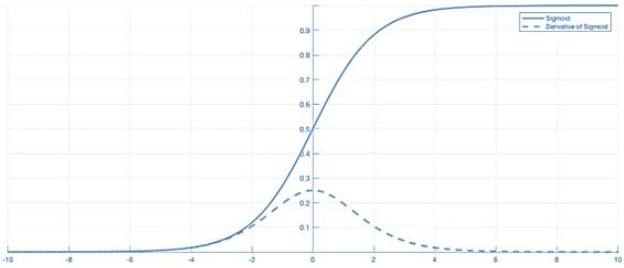


Figure 6. Graph of sigmoid function with its derivative.

The most important limitation that the sigmoid function creates in neural networks is the gradient vanishing problem, which occurs due to the derivative of this function. According to the image of the derivative curve of the sigmoid function, the gradient value of this function is very small for values greater than 3 or less than -3. When the gradient value tends to zero, the network stops learning [7].

- **ReLU activation function:**

ReLU activation function stands for Rectified Liner Unit. ReLU function is the default activation function used in deep learning [15,21,34]. This function is the most popular activation function thanks to its simple implementation, good performance and optimal effectiveness, and it is defined as follows:

$$f(x) = \max(0.x)$$

This function considers negative values as zero and positive values and values equal to zero as its own value. The derivative of the ReLU function is considered as follows:

$$f'(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$$

Note that when the input value is exactly equal to zero, the ReLU function is not derivable, and by default we consider the left derivative, i.e. zero value.

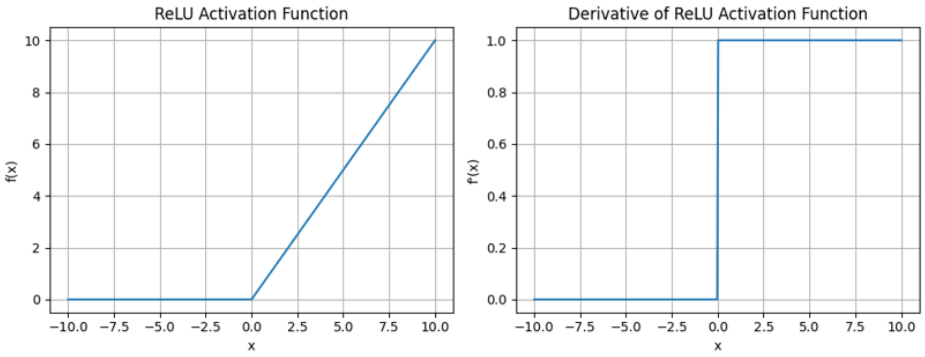


Figure 7. Graph of Rectified Liner Unit function with its derivative.

The ReLU activation function is a cheap computational function and it converges very quickly. In addition, it greatly accelerates the convergence of the decreasing gradient compared to the Sigmoid and Tanh activation function and can deal with the gradient vanishing problem [33]. When most of the inputs to the RELU activation function are in the negative range, they cause the decay problem. In the case that most of the outputs of the functions become zero, in the post-propagation stage, the gradients do not flow along the network and thus, the weights of the network are not updated.

3.2. *Parameterized standard activation functions*

This category is referred to as activation functions to all functions with a form very similar to a given constant form function, but having a set of learnable parameters that allow this shape to be set [11,12,31]. Adding these parameters to standard activation functions, and it requires changes in the learning process. For example, when using gradient based methods, partial derivatives of these new parameters are required. In the rest of this section, the first attempt to have a trainable activation function with the parameter presented in [17]. The proposed activation function was a generalization of the classical sigmoid function and by adding two parameters to adjust the shape of the activation function, i.e.:

$$\boldsymbol{Adjustable\ Generalized\ Sigmoid}(x) = \boldsymbol{AG}Sig(x) = \frac{\beta}{1+e^{-\alpha x}}$$

Both parameters are learned concurrently using gradient descent based on the backpropagation algorithm, which calculates the error function's derivatives with respect to network parameters. Another generalization of this function that used only one parameter was presented in [7,29] as follows:

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

These functions are parameterized by a value $m \in (0, +\infty)$.

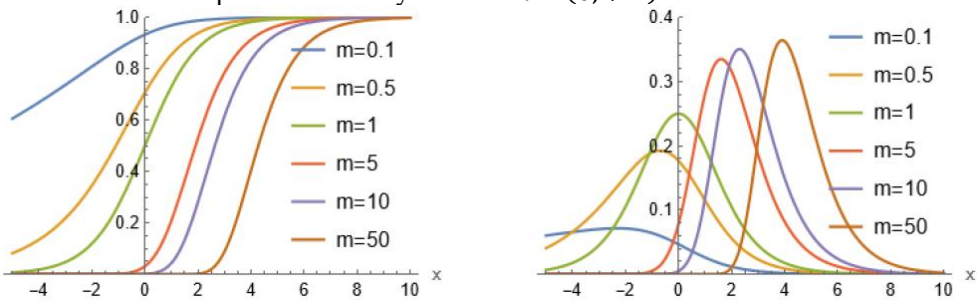


Figure 8. The figure illustrates the parametric Sigmoid for various values of $m \in$ (0.1, 0.5, 1, 5, 10, and 50) with its derivative.

Trottier (2017) [31] proposes an alternative ELU (exponential linear units) unit formulation with two learnable parameters, aiming to eliminate manual tuning of the ELU parameter, i.e.

$$PELU(x) = \begin{cases} \frac{\beta}{\alpha}x & if\ x \geq 0 \\ \beta.(e^{x/\alpha} - 1) & if\ x < 0 \end{cases}$$

The parameters α ,β > 0 regulate the function's shape and are jointly learned with other network parameters through a gradient-based optimization method.

Qiu (2018) [25] propose the following function:

$$Flexible\ ReLU\ \ = FReLU(x) = ReLU(x + \alpha) + \beta$$

with α , β learned by data. This modification aims to capture lost negative information in the classic ReLU function, while also preserving its zero-like property. This modification aims to capture lost negative information in the classic ReLU function, while also preserving its zero-like property. These efforts are still of interest to researchers today due to the different types of neural network models [25].

We focus on the Swish parametric activation function, which performed better than other activation functions in our neural network modeling.

3.3. *Swish*

In 2018, the swish activation function was introduced for the first time [4,26], and it is defined as follows:

$$f(x) = x * sigmoid(x) = \frac{x}{1 + e^{-x}}$$

This function is designed to be a more effective and efficient alternative to traditional activation functions such as ReLU and its variants. By adding the training parameter to this activation function, its new form is as follows:

$$Swish(x) = x\ Sigmoid(\beta x) = \frac{x}{1 + e^{-\beta x}}$$

- If β=1, the Swish function becomes a linear sigmoid function or SiLU [22].
- In β→+∞ mode, the function becomes ReLU or max (x.0).
- In the case of β→0, the resulting function is linear and equal to x/2.
- In β→-∞ mode, the function becomes min (x.0).

Like the ReLU activation function, the Swish activation function is bounded from above and bounded from below. Unlike the ReLU activation function, the Swish activation function is smooth and non-uniform. In fact, the difference between this function and the most common activation function is the non-uniformity of the Swish activation function, which helps to learn the weights of the network. The derivative of the Swish activation function is defined as follows:

$$\begin{aligned} f'(x) &= \sigma(\beta x) + \beta x \cdot \sigma(\beta x)\big(1 - \sigma(\beta x)\big) \\ &= \sigma(\beta x) + \big(\beta x \cdot \sigma(\beta x)\big) - \beta x \cdot \sigma(\beta x)^2 \\ &= \beta x \times \sigma(\beta x) + \sigma(\beta x)\big(1 - \beta x \cdot \sigma(\beta x)\big) \\ &= \beta f(x) + \sigma(\beta x)(1 - \beta f(x)) \end{aligned}$$

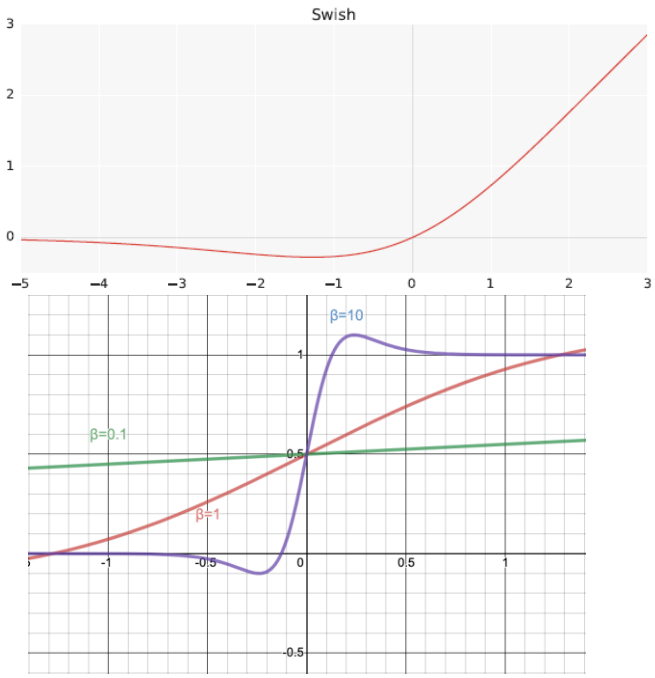


Figure 9. Graph of Swish function with various values of parameters.

The first and second derivatives of the Swish function are shown in the figure below. For inputs less than about 1.25, the value of the derivative is less than 1. A successful swish means that the gradient is preserved.

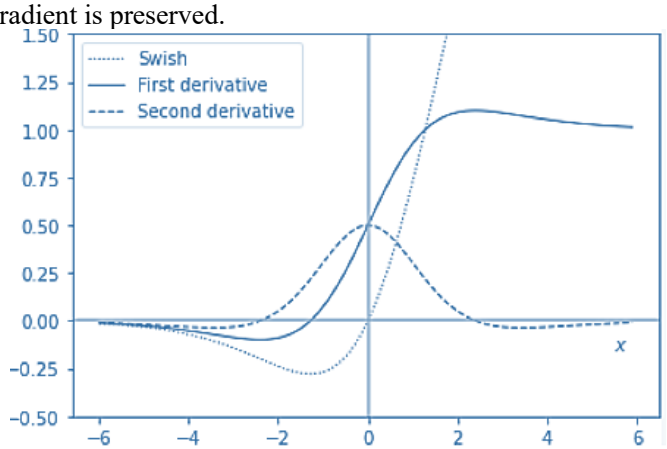


Figure 10. The first and second derivatives of the Swish function**.**

The analysis and examination of the advantages and disadvantages of the swish activation

function compared to other activation functions is given in [2]. Here we briefly discuss some of the differences between this activation function and the ReLU function

- **Advantages and disadvantages of Swish activation function:**

One of the advantages of the Swish activator function is the continuity of the function and its derivability along with the gentle and controllable slope of the curve. Unlike traditional activation functions such as ReLU, which have sharp turns or kinks, Swish is smooth and non-uniform. This smoothness helps to learn complex functions and gradient descent process during backpropagation. In addition; Unlike the ReLU activation function, it does

not suddenly change direction when the value of x is close to zero. In the ReLU activator function, negative values are converted to zero; While these negative values may contain important information of data patterns. In the Swish function, only large negative values are converted to zero.

The main drawback of the Swish activation function is that because the Swish function uses the sigmoid function in its calculations; The calculations of this function are heavier than the ReLU activation functions. Another weakness of this function is that based on the investigations, the performance of this function is unstable and its output cannot be predicted in advance.

4. Performance comparison of an experimental architecture

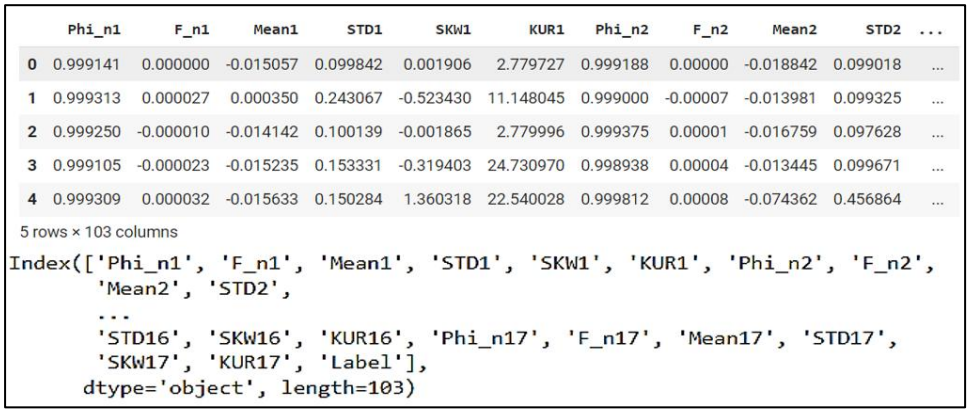
In this section, by using the modeling of a deep neural network, an attempt is made to detect the identity change of authorized devices from unauthorized devices to enter the local network and to prevent unauthorized devices from entering this secure network. Artificial intelligence algorithms can be used for preprocessing and analysis of RF signals[3]. In addition, AI-based RF analytics can detect and prevent unauthorized access or malicious activity in wireless networks, increasing security and reducing the risk of cyber-attacks [35]. Overall, the use of artificial intelligence in RF fingerprinting has the potential to significantly improve the accuracy and efficiency of RF signal identification and classification, enabling a wide range of applications in wireless communications, cyber security, and electronic warfare [19,23]. In this section, we demonstrate the performance of parametric activation functions on an experimental database that deals with the authentication of IoT devices. The following sections will describe our experimental setup and results in more detail.

4.1. *Modeling an experimental problem with a neural network*

The experimental problem in this article is of supervised learning and classification. In order to design the deep neural network of this article, firstly, we have considered the database containing the data set with 8000 samples and including 102 features and one label as the main data. This data was collected by One RF Hack radio receiver in a building with 24 rooms and on 4 Internet of Things devices. In a short time (3 seconds), this device records the number of 102 features along with a label that indicates the type of device in that particular situation. The database was created by a library named PyRadio in Python programming language to record and process wireless radio frequency signals at different points and 102 features were extracted and labeled to the desired device. These data have been saved in the form of a .csv file and various stages of data processing have been performed on it by the Scikit_learn library. TensorFlow platform and Functional API have been used for network architecture.

- **Neural network architecture process**

Neural network training involves adjusting the model parameters to minimize the difference between the network output and the desired output. The input data used to build the model are usually divided into multiple data sets. In particular, three data sets are usually used in different stages of model creation: training, validation, and test sets. At first, the model is built on a training data set in order to provide the ability to fit the parameters of the model by using a set of examples. It is then trained using a supervised learning method, such as optimization methods, on the training data set. Subsequently, the model built on the dataset is used to predict the results of the observations on a second dataset called the validation dataset. Finally, the test dataset is the dataset used to provide an unbiased evaluation of the final model fit to the training dataset.



	Phi_n1	F_n1	Mean1	STD1	SKW1	KUR1	Phi_n2	F_n2	Mean2	STD2	...
0	0.999141	0.000000	-0.015057	0.099842	0.001906	2.779727	0.999188	0.00000	-0.018842	0.099018	...
1	0.999313	0.000027	0.000350	0.243067	-0.523430	11.148045	0.999000	-0.00007	-0.013981	0.099325	...
2	0.999250	-0.000010	-0.014142	0.100139	-0.001865	2.779996	0.999375	0.00001	-0.016759	0.097628	...
3	0.999105	-0.000023	-0.015235	0.153331	-0.319403	24.730970	0.998938	0.00004	-0.013445	0.099671	...
4	0.999309	0.000032	-0.015633	0.150284	1.360318	22.540028	0.999812	0.00008	-0.074362	0.456864	...

5 rows × 103 columns

```
Index(['Phi_n1', 'F_n1', 'Mean1', 'STD1', 'SKW1', 'KUR1', 'Phi_n2', 'F_n2',
       'Mean2', 'STD2',
       ...
       'STD16', 'SKW16', 'KUR16', 'Phi_n17', 'F_n17', 'Mean17', 'STD17',
       'SKW17', 'KUR17', 'Label'],
      dtype='object', length=103)
```

Figure 11. Experimental data table in TensorFlow platform**.**

We define the model using TensorFlow and Cross libraries. The number of training data in the model was determined to be 6480, equivalent to 70% of the data set. Also, the number of test and evaluator data is 800, equivalent to 15% of the total data set, and the number of evaluator data is 720, equivalent to 15% of the training data. Each sample has 102 features as neural network inputs and a label as output. It should be noted that the training and test data in this model have the same distribution and the aggregation of the test data was not done in a specific area of the training samples. Also, a specific sample of devices is not used to measure the performance of the model, and among the 4 types of devices in the network training process, there is a test sample among the test data. Figure 12 shows these claims. We train fully connected networks of different depths on this database, each layer having 20 to 40 neurons.

The presented deep neural network architecture is the result of examining a large number of different types of hyperparameters, including the application of different methods in the initial weighting of layers, changes in the learning rate of the Adam optimizer algorithm [13], changes in the number of hidden layers, neurons and the number of repetitions of the network and specifically the types of activation functions. The network is optimized using Adam on a batch size of 256, and for fair comparison, we try the same number of learning rates for each activation function.

In the future model, using the possibility of grid search, the appropriate range for the hyperparameters of the model, the learning rate is in the range of 0.0001, the number of neurons is around 20 to 40, and the number of hidden layers is between 3 and 5. is provided. All networks are initialized with He initialization (He et al., 2015) [15]. The Python codes related to the network architecture are located at the GitHub address https://github.com/SaeidSarabadan/RF_with_-ANN.

4.2. *Comparison of activation functions on the model*

The following figure shows the test accuracy when changing the number of layers in the model with different activation functions on the experimental problem. This figure shows the average result of 3 runs.

In our experiments, activation functions up to 4 layers have almost the same performance. However, as shown in the figure, Swish outperforms the other activation functions by a large margin in the range between 3 and 4.

We compare Swish with several parametric activation functions in this experimental model and using the grid search process in the parameters of the activation functions, we have reached the results that we have given in Table 15. Since there are many parametric activation functions, we have selected the most common ones as follows:

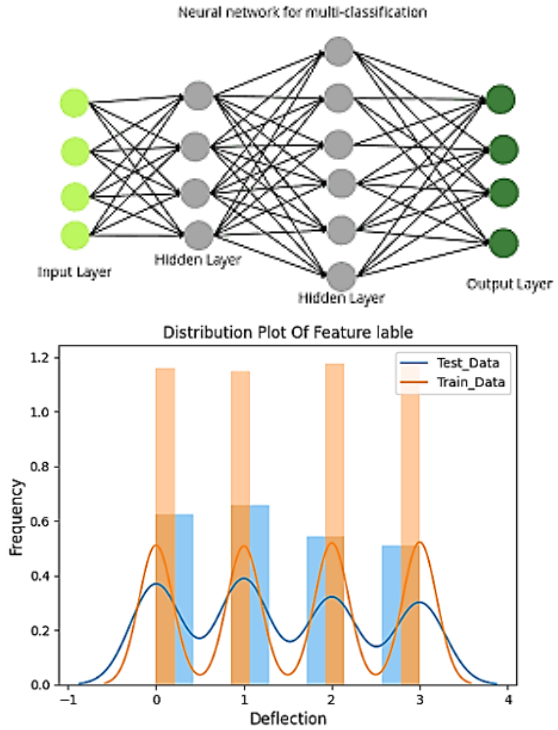


Figure 12: Distribution plot of Feature lable.

- Leaky ReLU (LReLU)[21] (Maas et al., 2013):

$$f(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases} \quad \text{Where } \alpha = 0.868.$$

- Scaled Exponential Linear Unit [20] (SELU) (Klambauer et al., 2017):

$$\text{SELU}(x) = \lambda \begin{cases} \alpha(e^x - 1). & \text{for } x < 0 \\ x. & \text{for } x \geq 0 \end{cases} \quad \text{Where } \alpha \approx 1.8683 \text{ and } \lambda \approx 1.0483 .$$

- Parametric ReLU [15] (PReLU) (He et al., 2015):

The same form as LReLU but α is a learnable parameter. Each channel has a shared α which is initialized to 0.12.

- Smish [32] (Achuan Wang et al., 2022):

$$Mish(x) = x \cdot tanh\,(ln(1 + e^x))$$
$$Smish(x) = x \cdot tanh\,(ln(1 + \text{sigmoid}(x))$$

The Smish function aligns with the Logish function in its early stages by using the sigmoid function to reduce value ranges and applying a logarithmic operation to create a smooth curve and stable trend [9,12].

- Swish[26] (Ramachandran, P , 2018]):

by adding a multiplicative coefficient to the SiLU function, and obtaining:

$$Swish(x) = x\,Sigmoid(\beta x) = \frac{x}{1 + e^{-\beta x}}$$

With $\beta = 1.852$

Here, due to the fact that the wish activation function has a higher efficiency in this experimental example, its network architecture specifications are given in the table below, and this table has been omitted for other activation functions used.

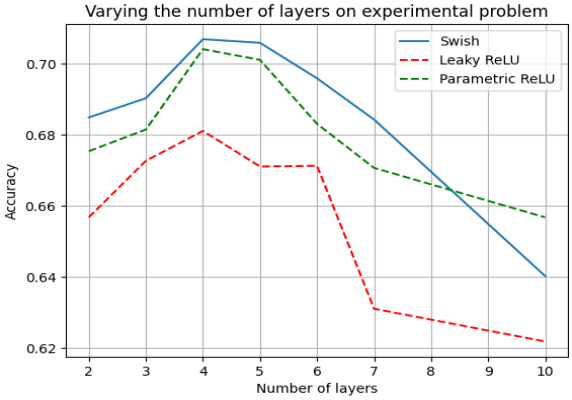


Figure 13.The test accuracy when changing the number of layers in fully connected networks with different activation functions.

Table 1. Performance of Swish activation function on experimental dataset.

Swish activation function. **Loss function: Categorical Cross Entropy. Optimizer algorithm is ADAM with learning rate=0.0001 ،$\beta_2 = 0.998$ ،$\beta_1 = 0.97$.** **The number of epochs is 3000.**									
Components of deep neural networks			**Model evaluation indicators**				**Model complexity**		
Number of hidden layers	**Number of neurons**	**β**	**Recall**	**Precision**	**accuracy**	**F1_score**	**GPU used**	**Model execution time**	**Number of parameters**
2	20,40	1.852	0.7545	0.7895	0.7560	0.7522	0.3	461.7	3064
3	20,40,20	1.852	0.7610	0.7616	0.7641	0.7612	0.3	474.9	3804
4	10,20,30,40	1.852	0.7814	0.7845	0.7895	0.7822	0.3	388.5	4384

As mentioned before, we have considered 15 percent of the training data in order to avoid overfitting the model, whose We plot the learning curves for this model with swish activation function in Figure 14.

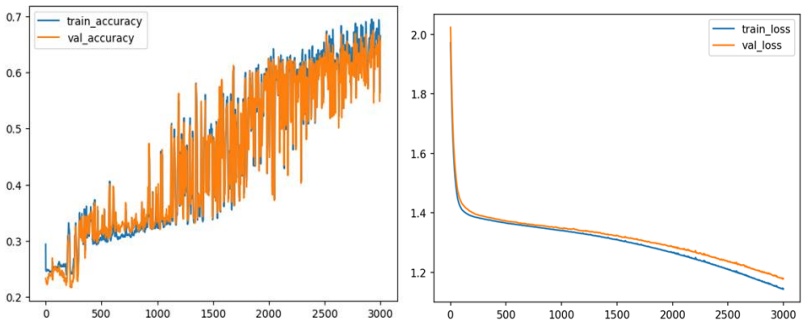


Figure 14. Training curves of Model.

- Performance comparison of an experimental architecture

We have compared all the activation functions in various designed architectures and on the collected authentication database. For each activation function, we try 3 different learning rates with ADAM and pick the best. To verify that the performance differences are reproducible, we run the Models experiments 3 times with the best learning rate from the first experiment. The results in Tables 2 show strong performance for Swish.

Table 2. Performance of Multiple activation functions on experimental dataset.

ReLU ،Swish ،LReLU ،PReLU و SELU Loss function: Categorical Cross Entropy. Optimizer algorithm is ADAM with learning rate=0.0001 ، $\beta_2 = 0.998$ و$\beta_1 = 0.97$. The number of epochs is 3000.The hidden layer :20,30,40,10 with 4384 parameters.							
Components of deep neural networks		**Model evaluation indicators**				**Model complexity**	
Number of neurons	**Parameteriz activation functions**	**Recall**	**Precision**	**accuracy**	**F1_score**	**GPU used**	**Model execution time**
RELU(2010)	_	0.6912	0.6991	0.7069	0.6951	0.3123	345.3
Leaky ReLU (2013)	0.0158	0.6894	0.6885	0.6811	0.6893	0.3254	522.6
Parametric ReLU(2015)	0.23	0.6981	0.6997	0.7041	0.6996	0.389	501.8
SELU (2017)	$\alpha = 1.86$ $\lambda = 1.04$	0.7183	0.7140	0.7215	0.7179	0.3921	459.6
Mish (2019)	_	0.7565	0.7569	0.7561	0.7568	0.3628	478/9
Swish (2018)	1.852	0.7814	0.7845	0.7895	0.7829	0.3876	388.5

Swish also matches or exceeds the best performing baseline on most models, where again, the best performing baseline differs depending on the model. Our study shows that Swish consistently outperforms ReLU, Parametric ReLU, SELU, on this model.

5. Conclusion

In this work, we improved the architecture of a deep neural network aimed at authenticating IoT devices by using different parametric activation functions. Among the most important results obtained in this article, we can mention the reduction of the error rate and time of model training and the increase of accuracy and speed of model training with the changes of the activator functions. According to our experimental test, the swish activation function was selected as the best option by determining the appropriate parameter. The use of convolutional networks in this research was not considered due to the type of database and having simple and one-dimensional numerical data, but it is recommended for modeling with larger data and higher dimensions.

Authentication in the Internet of Things is one of the most important and up-to-date issues, and it is done by different methods, and the use of neural networks is considered one of the effective methods. The database used is in a private location and collected by the One RF Hack radio receiver device.

References

[1] Abayneh Bezabih, Mathematical Modeling of COVID-19 Pandemic with Treatment, International Journal of Mathematical Modelling & Computations, 11 (2021), doi:10.30495/ijm2c.2021.684815.

[2] F. Agostinelli, M. Hoffman, P. Sadowski and P. Baldi, Learning Activation Functions to Improve Deep Neural Networks, arXiv preprint arXiv:1412.6830 (2014), doi:10.48550/arXiv.1412.6830.

[3] S. Al-Hazbi, A. Hussain, S. Sciancalepore, G. Oligeri and P. Papadimitratos, Radio Frequency Fingerprinting via Deep Learning: Challenges and Opportunities, arXiv preprint arXiv:2310.16406v2 (2024), doi:10.48550/arXiv.2310.16406.

[4] E. Alcaide, E-swish: Adjusting Activations to Different Network Depths, arXiv (2018), doi:10.48550/arXiv.1801.07145.

[5] A. Apicella, F. Donnarumma, F. Isgrò and R. Prevete, A Survey on Modern Trainable Activation Functions, Neural Networks, 138 (2021) 14-32, doi:10.1016/j.neunet.2021.01.026.

[6] C. M. Bishop and H. Bishop, Deep Learning Foundations and Concepts, Springer, (2023).

[7] P. Chandra and Y. Singh, An Activation Function Adapting Training Algorithm for Sigmoidal Feedforward Networks, Neurocomputing, 61 (2004) 429-437, doi:10.1016/j.neucom.2004.04.001.

[8] P. Cheridito, A. Jentzen, A. Riekert and F. Rossmannek, A Proof of Convergence for Gradient Descent in the Training of Artificial Neural Networks for Constant Target Functions, Journal of Complexity, 101646 (2022), in press, doi:10.1016/j.jco.2022.101646.

[9] H. H. Chieng, N. Wahid, O. Pauline and S. R. K. Perla, Flatten-tswish: A Thresholded ReLU-Swish-like Activation Function for Deep Learning, International Journal of Advances in Intelligent Informatics, 4 (2) (2018) 76-86, doi:10.26555/ijain.v4i2.249.

[10] A. Dureja and P. Pahwa, Analysis of Non-linear Activation Functions for Classification Tasks Using Convolutional Neural Networks, Recent Patents on Computer Science, 12 (2019) 156-161, doi:10.2174/2213275911666181025143029.

[11] S. Elfwing, E. Uchibe and K. Doya, Sigmoid-weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning, Neural Networks, 107 (2018) 3-11, doi:10.1016/j.neunet.2017.12.012.

[12] L. B. Godfrey, An Evaluation of Parametric Activation Functions for Deep Learning, IEEE International Conference on Systems, Man and Cybernetics (SMC), (2019) 3006-3011, doi:10.1109/SMC.2019.8913972.

[13] Hossein Abbasiyan and Mohammad Jafar Doostideilami, Two methods to obtain preferred efficiency for negative data (IS), International Journal of Mathematical Modelling & Computations, 13 (2023), doi:10.30495/ijm2c.2023.1952616.1246.

[14] M. M. Hammad, Statistics for Machine Learning with Mathematica Applications, arXiv (2023), doi:10.48550/arXiv.2310.00004.

[15] K. He, X. Zhang, S. Ren and J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Proceedings of the IEEE International Conference on Computer Vision, (2015) 1026-1034, doi:10.1109/ICCV.2015.123.
[16] X. Hu, W. Liu, J. Bian and J. Pei, Measuring Model Complexity of Neural Networks with Curve Activation Functions, Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (2020) 1521-1531, doi:10.1145/3394486.3403203.
[17] Z. Hu and H. Shao, The Study of Neural Network Adaptive Control Systems, Control and Decision, 7 (2) (1992) 361-366.
[18] A. Jagannath, J. Jagannath and P. S. P. V. Kumar, A Comprehensive Survey on Radio Frequency (RF) Fingerprinting: Traditional Approaches, Deep Learning, and Open Challenges, arXiv preprint arXiv:2201.00680v3 (2022), doi:10.48550/arXiv.2201.00680.
[19] T. Jian, B. C. Rendon, E. Ojuba et al., Deep Learning for RF Fingerprinting: A Massive Experimental Study, Northeastern University, Boston, MA, USA, (2020), doi:10.1109/IOTM.0001.1900065.
[20] G. Klambauer, T. Unterthiner, A. Mayr and S. Hochreiter, Self-normalizing neural networks, Advances in neural information processing systems, 30 (2017), doi:10.48550/arXiv.1706.02515.
[21] A. L. Maas, A. Y. Hannun and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, Proc. icml, 30 (1) (2013) 3.
[22] F. Makhdoom and J. Ul Rahman, A Comparative Exploration of Activation Functions for Image Classification in Convolutional Neural Networks, i-Manager's Journal on Artificial Intelligence & Machine Learning, 2 (1) (2024), doi:10.26634/jaim.2.1.20225.
[23] A. Pirnat, B. Bertalanič, G. Cerar et al., Towards Sustainable Deep Learning for Wireless Fingerprinting Localization, arXiv preprint arXiv:2201.09071v1 (2022), doi:10.48550/arXiv.2201.09071.
[24] S. J. D. Prince, Understanding Deep Learning, MIT Press, Cambridge, MA, (2023).
[25] S. Qiu, X. Xu and B. Cai, Frelu: Flexible Rectified Linear Units for Improving Convolutional Neural Networks, 2018 24th International Conference on Pattern Recognition (ICPR), (2018) 1223-1228, doi:10.1109/ICPR.2018.8546022.
[26] P. Ramachandran, B. Zoph and Q. V. Le, Searching for Activation Functions, 6th International Conference on Learning Representations, (2018), doi:10.48550/arXiv.1710.05941.
[27] S. K. Roy, S. Manna, S. R. Dubey and B. B. Chaudhuri, LiSHT: Non-parametric Linearly Scaled Hyperbolic Tangent Activation Function for Neural Networks, Computer Vision and Image Processing. CVIP 2022. Communications in Computer and Information Science, 1776 (2023) 445-457, doi:10.48550/arXiv.1901.05894.
[28] S. Scardapane, S. Van Vaerenbergh, S. Totaro and A. Uncini, Kafnets: Kernel-based Non-parametric Activation Functions for Neural Networks, Neural Networks, 110 (2019) 19-32, doi:10.48550/arXiv.1707.04035.
[29] Y. Singh and P. Chandra, A Class + 1 Sigmoidal Activation Functions for FFANNs, Journal of Economic Dynamics and Control, 28 (1) (2004) 183-187, doi:10.1007/s11761-024-00423-w.
[30] G. Strang, Linear Algebra and Learning from Data, Wellesley-Cambridge Press, (2019).
[31] L. Trottier, P. Gigu, B. Chaib-draa et al., Parametric Exponential Linear Unit for Deep Convolutional Neural Networks, Machine Learning and Applications (ICMLA), 16th IEEE International Conference on (2017) 207-214, doi:10.1109/ICMLA.2017.00038.
[32] X. Wang, H. Ren and A. Wang, Smish: A novel activation function for deep learning methods, Electronics, 11 (4) (2022) 540, doi:10.3390/electronics11040540.
[33] B. Xu, N. Wang, T. Chen and M. Li, Empirical Evaluation of Rectified Activations in Convolutional Networks, arXiv (2015), doi:10.48550/arXiv.1505.00853.
[34] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang and S. Peng, Rectified Exponential Units for Convolutional Neural Networks, IEEE Access, 7 (2019) 101633-101640, doi:10.1109/ACCESS.2019.2928442.
[35] J. Zhang, G. Shen, W. Saad and K. Chowdhury, Radio Frequency Fingerprint Identification for Device Authentication in the Internet of Things, IEEE Communications Magazine, (2023), doi:10.1109/MCOM.003.2200974.