# 

# **CHAPTER 1**

1. [**Introduction And Motivation**](about:blank)

Thanks to significant developments in CMOS technology in design and implementation of Microsystems, Artificial Neural Networks (ANNs) have shown improvements in performance in a wide range of applications including speech recognition [17], signal processing [?], robotics [?], computer vision [?], natural language processing [?], medicine [?], and automatic driving [68]. ANNs have been designed to mimic the functions of the human brain that learn from experiences and adapt accordingly to the situation. Like the human brain which has a multi-tiered structures which are containing billions of neurons arranged in a hierarchy layer, ANNs also have a network of neurons that are interconnected to each other via axons [68].

ANNs were typically implemented only in software such as CPUS and GPUS for a couple of decades. Software implementation had a drawback of long running times. ANNs hardware implementation and ANN accelerators have become popular recently because of their hardware performance and processing speed. Hardware implementations have improved dramatically due to the inherently parallel structure of Field Programmable Gate Arrays (FPGAs) [21–23] with low-cost and power architectures for NNs implementation [68] and Application Specific Integrated Circuits (ASICs) [18–20]. The modularity and reconfigurability character of FPGAs and ASICs makes them popular in comparison to the software implementations. However, special attention should be paid when we design the ANN in hardware because the nonlinear activation function is the most critical and hardest components to implement.

* 1. [***Neural Network***](about:blank)**s**

There is a simple perceptron in figure 1, and each input like x1, x2, …, xn would be multiplied to their own (synaptic) weights then an activation function will be applied to convert the linear input signals of a node into non-linear output signals to facilitate the learning of high order polynomials that go beyond one degree for deep networks. The output will be compared to expected result and the difference will be the absolute error.

There is a simple perceptron in figure 1, and every input has been multiplied to their own weights then activation function has been applied in order to convert linear output to non-linear for facilitating training model. The output will be compared to the expected result and the difference will be called as error.

The next process will be backward propagation which includes inverse direction from output towards input and compute differentiation of activation function for optimal point. Optimal point leads to having minimum error which is the fundamental desire in the NNs. Each back and forth or forward and backward propagation has been named an epoch, the sum of epochs nominates total iteration in NNs.

The next process is backward propagation, which performs its operation from output to input. The derivative of the activation function has been computed for finding the optimal point. The role of optimal point is to obtain minimum error. The minimum error is the fundamental goal in the neural network for having the best precision. Every iteration for computing new error and weight includes forward and backward propagation is called as epoch.

Diagram

Description automatically generated

*Figure 1.1: Neural Network and Neurons, Weights, Activation Functions*

Activation function, both digital and analog, is used as the output of each neuron depending on the architecture of ANN. Every activation function takes a single value and performs a certain fixed mathematical operation on it. These functions are used between each layer to the output values from the previous layer to the endurable range of input values for the next layer. For this reason, a variety type of activation functions has been used in neural networks.

Activation functions are linear (ReLU) or none-linear (Sigmoid, Tanh). Activation functions are applied on neurons takes a value and tries to curve and round output by the aid of certain mathematical operations.

The activation function is one of the most significant components which introduce non-linearity into the learning process in each neural network. While many activation functions are needed in the hardware implementation of a neural network, a small reduction in hardware resources can cause a dramatic decrease in area usage and resource consumption [20]. However, studies show that accurate nonlinear activation function implementations can improve the learning and generalization abilities of ANNs. Architectures with higher accuracy will increase silicon area (resources) usage and reduce the computational speed. Accordingly, having nonlinear activation function hardware design with high speed, and acceptable range of accuracy, and small areas has become the ideal approach for every hardware implementation of neural networks.

The activation function is one of the most significant components which introduce non-linearity into the learning process in each neural network. Activation functions play important role in neural networks which cannot be neglected, their existence can guarantee the precision especially where there is large amount of data. Implementing activation functions in FPGA must be efficiently, otherwise causes heavy hardware cost with so many basic logic gates. However, studies show that accurate nonlinear activation function implementations can improve the learning and generalization abilities of neural network. Efficient architectures will increase resources performance and reduce the computational speed. Accordingly, having nonlinear activation function hardware design with high speed, acceptable accuracy, and less resources has become the ideal approach for neural networks.

Several different activation functions are available today, including sigmoid, Elliot, Tanh, RELU, soft-max, SWISH, Parametric RELU, and many more [32]. Because of zero-hard rectification, some of the existing activation functions such as RELU and Swish fail to utilize the large negative input values and may suffer from the dying gradient problem, which should be tackled.

?????????????????????????

Many studies consider hardware implementations of activation functions. They focus on various aspects such as cost of implementation, accuracy, applied approximation methods, and analog or digital bases. Thus, it is important to look for a better activation function according to the structure of the neuron network [1].

* + 1. [**Activation function**](about:blank)***s***

The vital requirement of a neural network is to have a better prediction and higher precision in the results. So, the nonlinear functions have more rounding and bending which can categorize clusters more precisely. In the comparison between linear and nonlinear functions, it is obvious that nonlinear is more accurate to predict and has a better boundary decision line for categorizing and classification. Figure 1.2 shows a linear function that could not separate one of the red circles while with the aid of nonlinear function at the right picture all the red circle and a blue triangle could be separated perfectly. Activation functions as exponential or tangency function would be applied on the NNs through forward and backward propagation to find better weight which leads to having less error.

Ideal neural networks have accurate prediction and higher precision in the results. So, the nonlinear functions with rounding and curving shape, have more flexibility to categorize clusters precisely. In the comparison between linear and nonlinear functions, it is obvious that nonlinear is more accurate to predict and has a better boundary decision line for categorizing and classification. Figure 1.2 shows a linear function that could not separate one of the red circles, while with the aid of a nonlinear function at the right picture all the red circle and a blue triangle could be separated perfectly. Exponential or tangential activation functions navigate throughout forward and backward propagation to find better weight, which leads to having less error.

*Chart, line chart

Description automatically generated*

*Figure1.2. Linear and Nonlinear Functions.*

These sorts of activation functions must be differentiable because backward propagation needs to compute the better weights and answers and it needs to calculate the minimum optimal point in this process. figure 1.3 shows a list of well-known activation functions such as sigmoid, tangent hyperbolic (Tanh), SoftMax, and Rectified Linear Unit (ReLU), each of these activation functions have their own cons and pros.

The first step to improve precision is to compute the differentiation from activation function because backward propagation needs to compute the better weights, so it needs to calculate the optimal point in this process. Figure 1.3 shows a list of well-known activation functions such as sigmoid, tangent hyperbolic (Tanh), SoftMax, and Rectified Linear Unit (ReLU), each of these activation functions have their own cons and pros.

Table

Description automatically generated

*Figure 1.3. List of Activation Functions [69].*

### *Rectified Linear Unit (ReLU)*

ReLU is a fast-learning activation function that gives the best performance and results, due to its own generalization and as this thesis mentioned, generalization is the uppermost and elegant feature of active functions to reduce time consumption and enhance precision.

ReLU is a fast-learning activation function that gives the best performance and results. Time consumption reduction and higher precision are the result of generalization and elegant feature of active functions.

ReLU is a nearly linear function that retains the properties of linear models, which makes them easy to optimize with gradient-descent methods. ReLU activation function guarantees a faster computation – it does not compute exponentials and divisions, thereby boosting the overall computation speed.

The ReLU function performs a threshold operation on each input element where all values less than zero are set to zero. Thus, the ReLU is represented as:

|  |  |
| --- | --- |
|  | (1.1) |

### *Sigmoid*

The sigmoid function is a non-linear activation function in feedforward neural networks. It is a differentiable real function and has a specific degree of smoothness. The sigmoid function appears in the output layer of the NNs models and is used for predicting probability-based outputs. The sigmoid function is represented as:

~~used for predicting probability-based outputs.~~

|  |  |
| --- | --- |
|  | (1.2) |

Gradient saturation is a problem in the sigmoid function which means that after some epochs, the learning happens fast, but at some points, the value of the linear part will be far from the center of the sigmoid and its saturation. Indeed, gradient saturation takes too much time to update the weights because of the small value of its gradient.

Gradient saturation is a problem in the sigmoid function. After some epochs that learning happens relatively fast, the value of the linear part will be far from the center of the sigmoid, and it takes too much time to update the weights because the value of gradient is small.

Another drawback of the sigmoid function is slow convergence, sharp damp gradients during backpropagation from deeper hidden layers to the input layers, and non-zero-centered output that causes the gradient updates to propagate in varying directions.

### *Tangent Hyperbolic*

The hyperbolic tangent function is a smoother, zero-centered function having a range between -1 to 1. Fiure1.4 shows the output of the tanh function.

Diagram

Description automatically generated

*Figure1.4. Gradient Saturation x>-4 and x<-4.*

|  |  |
| --- | --- |
|  | (1.3) |

The tanh function delivers better training performance for multilayer neural networks rather than sigmoid function. The biggest advantage of the tanh function is that it produces a zero-centered output, for supporting the backpropagation process. The Tanh function has been mostly used in recurrent neural networks (RNN)[?] for natural language processing (NLP)[?] and speech recognition (SR)[?].

The biggest advantage of the tanh function is that it produces a zero-centered output, for supporting the backpropagation process. Because, all-positive or all-negative activation functions (relu, sigmoid) can be difficult for gradient based optimization.

The problem of Tanh is that it cannot solve the vanishing gradient problem. The gradient will be increased exponentially when NNs propagate the model until they eventually explode. Also, the Tanh function can only obtain a gradient of 1 when the input value is 0 (x is zero). Therefore, Tanh can produce some dead neurons during the computation process.

Among all different types of activation functions, Hyperbolic Tangent is chosen because of its ideal steep derivative that allows a wider range of values for fast learning and grading methods. Hyperbolic Tangent is easily differentiable and compatible with derivative-based learning approaches. Tanh is much better for learning than the sigmoid function [33] and became famous because of two specific features. Firstly, Tanh generates the output values from -1 to 1 and, more importantly, covers an entire positive to a negative range of inputs. Secondly, it is a symmetric function, which makes it easier to be implemented in hardware [3].

The Tanh and sigmoid functions both produce a curve with an “S” shape, where the Tanh output varies between [-1,1] and the sigmoid output varies between [0,1].

The Tanh function is defined as eq. 1.4, which is shown in Fig. 1.5. which saturates outside the range of [−4, 4].

Chart

Description automatically generated with medium confidence

Fig1.5. Hyperbolic Tangent & Sigmoid functions.

(1.4)

Sigmoid(x) =

To simplify the equations and find the relation between Tanh and Sigmoid (1.4) can be written as:

By dividing the numerator and denominator by ,(1.6) is changed to:

Despite the fact that analog circuits have some advantages in factors such as operation speed, but their sensitivity to noise, temperature changing, and power consumption variations is high. However, because digital circuits operate just at two voltages zero and one with the additional advantage of re-programmability. Motivated by this fact, in this dissertation, a digital architecture design of hyperbolic tangent function, is presented. A low complexity accurate hardware implementation of the activation function is required to meet the performance and area targets of the neural network accelerators.

Artificial Neural Networks (ANN) are popular computing models because of their precision and ability to work with large amounts of data. Many different fields of study use ANNs from instrumentation, measurements, and bioinformatics to digital signal processing.

Many fields of study use artificial neural network algorithms, such as bioinformatics and digital signal processing.

As mentioned above, ANNs include so many operations such as multiplications, additions, and subtractions. because in deep learning, ANNs have many hidden layers, and there are multiple activation functions in each layer finding the best weight for each neuron consequently leads to the best precisions in whole ANNs. The exponential part of a nonlinear activation functions causes hardware costs and high time consumption.

causes hardware costs and much time for processing.

ANNs are implemented in dedicated hardware, such as Field Programmable Gate Array (FPGA) or Application-Specific Integrated Circuits (ASIC), due to cost-effective and real-time results [40]. Computing directly on FPGA or ASIC with nonlinear activation functions or conventional arithmetic circuits is challenging because it involves many multiplications and additional gates which causes hardware cost and delay.

There are multiple approaches to prevent this dilemma and evaluate hardware implementation. The approaches to overcome this challenge are some approximation methods such as

Dilemma means the problem with having the large number of logical gate without below approaches.

* Lookup-Tables (LUTs),
* Range Addressable Lookup-Tables (RALUTs),
* Piece Wise Linear approximation (PWL),
* Power of Two, and
* Coordinate Rotation Digital Computer approximation (CORDIC).