Department of Computer Science
American International University-Bangladesh
Mid Term Report

Course Name: COMPUTER VISION AND PATTERN RECOGNITION

"A Report on Activation Function"

Supervised By:

DR. DEBAJYOTI KARMAKER

Associate Professor, Computer Science-AIUB

Submitted By:

Saima Sadia Ratri

ID: 20-43793-2

Section: B

Submission Date: March 11, 2023.

Activation Function:

Activation Functions are mathematical function specially used in artificial neural networks to transform an input signal into an output signal which in turn is fed as input to the next layer in the stack. The purpose of an activation function is to introduce nonlinearity into a neuron's output, which enables the network to simulate complex relationships between inputs and outputs.

Activation Function work methodology in Neural Networks:

When an input is fed into a neuron, the neuron performs a weighted sum of the input values and adds a bias term. This produces a single output value, which is then passed through the activation function. The activation function applies a mathematical operation to the output of the neuron, transforming it into a new value that is then passed to the next layer of neurons in the network. The specific operation performed by the activation function depends on the type of function being used. For example, the sigmoid function maps any input to a value between 0 and 1, which can be interpreted as a probability. The ReLU function, on the other hand, maps any input less than zero to zero and any input greater than zero to itself, which produces a sparse, nonlinear output. The choice of activation function depends on the specific problem and the architecture of the neural network. Different activation functions have different properties, and some are better suited to certain types of problems than others.

There are Different types of Activation Function, such as:

- Step Function
- Sigmoid Function
- TanH Function
- Relu Function
- Elu Function
- Selu Function

Step Function:

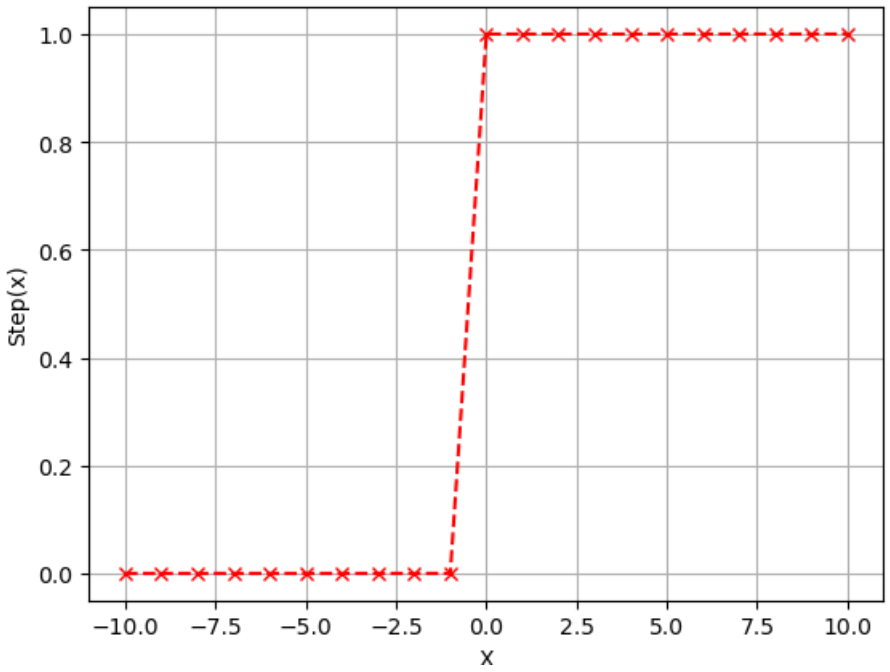
Step Function is one of the simplest kinds of activation functions. In this, we consider a threshold value and if the value of net input say y is greater than the threshold then the neuron is activated.

Mathematically,

f(x)=1, \text{if x}>=0

f(x)=0, \text{if x}<0

```
y = np.zeros(len(x))
y[x>=0] = 1
plot_graph(y, "Step(x)")
```



The step function is often used in binary classification problems, where the goal is to classify input data into one of two categories. For example, in a spam detection system, the input data might be an email message, and the two categories might be "spam" and "not spam". The step function can be used to map the output of a neural network to one of these two categories.

Sigmoid Function:

The sigmoid function is a popular activation function used in neural networks. It maps any input value to a value between 0 and 1, which can be interpreted as a probability. The sigmoid function is defined mathematically as:

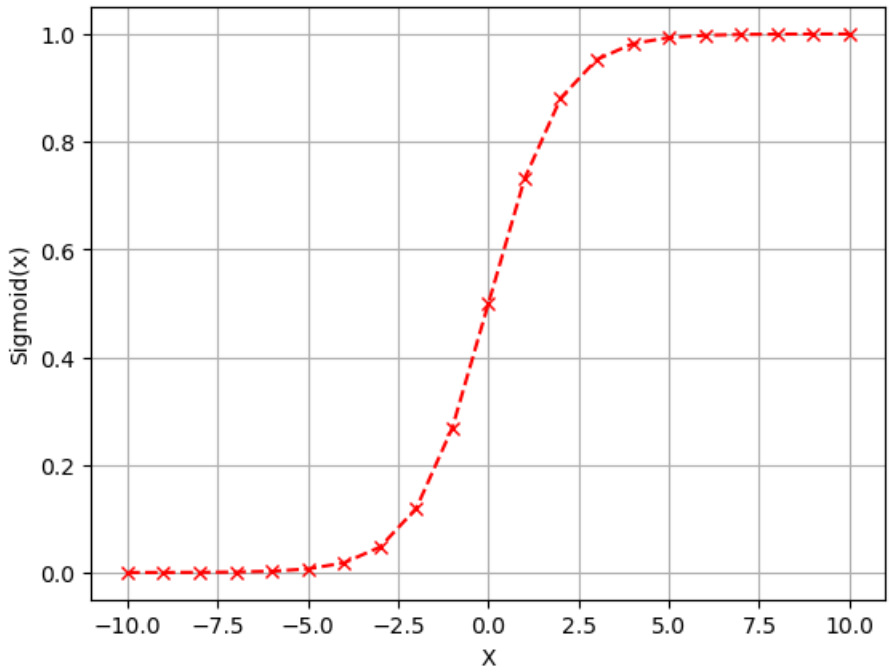
f(x) = 1 / (1 + e^(-x))

where x is the input to the neuron.

The sigmoid function has a characteristic S-shaped curve, which allows it to introduce nonlinearity into the output of the neuron. The steepness of the curve can be controlled by adjusting the value of the slope parameter. When the input to the sigmoid function is positive, the output is close to 1, and when the input is negative, the output is close to 0. When the input is close to 0, the output is close to 0.5.

```
y = 1/(1 + np.exp(-x))


plot_graph(y, "Sigmoid(x)")
```



The sigmoid function is often used in binary classification problems, where the goal is to classify input data into one of two categories. For example, in a medical diagnosis system, the input data might be a set of patient features, and the two categories might be "fit" and "sick". The output of the sigmoid function can be interpreted as the probability that the patient is sick, and a threshold value can be set to determine the final classification.

TanH Function:

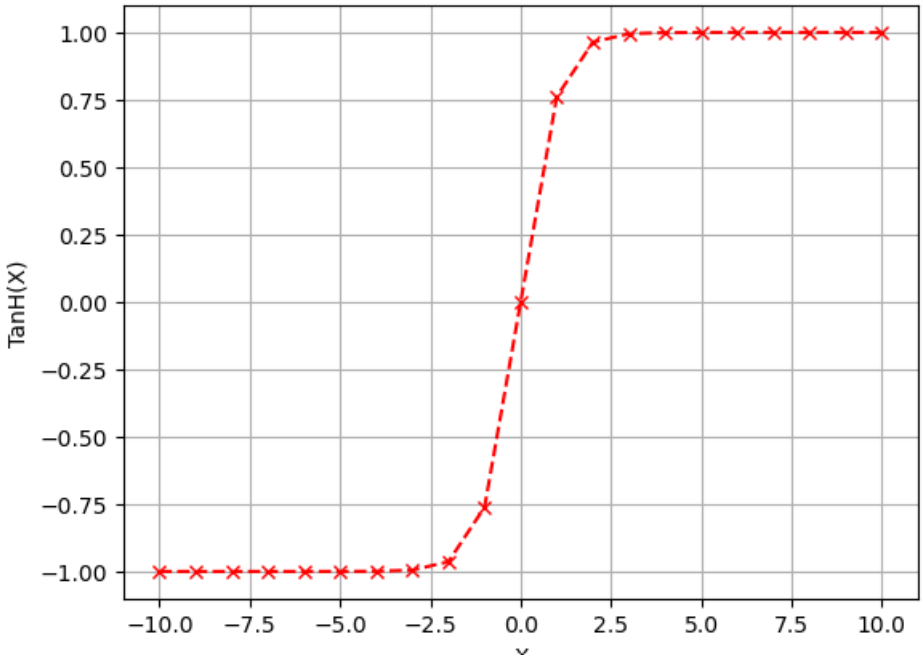
TanH Function maps any input value to a value between -1 and 1, which can be interpreted as a centered probability. The tanh function is defined mathematically as:

f(x) = (e^(x) - e^(-x)) / (e^(x) + e^(-x))

where x is the input to the neuron.

The TanH function has a similar shape to the sigmoid function but is centered at 0. Like the sigmoid function, the tanh function introduces nonlinearity into the output of the neuron. When the input to the tanh function is positive, the output is close to 1, and when the input is negative, the output is close to -1. When the input is close to 0, the output is close to 0.

```
y = (np.exp(2*x) - 1) / (np.exp(2*x) + 1)
plot_graph(y, "TanH(X)")
```



The TanH function is often used in neural networks as an alternative to the sigmoid function, as it can produce more stable and accurate results. It is especially useful in applications where the input data is standardized or normalized, as it can center the data around zero and produce outputs that are more interpretable.

However, like the sigmoid function, the TanH function can also suffer from the vanishing gradient problem. This occurs when the gradient of the tanh function approaches zero as the input to the neuron becomes very positive or very negative, which can cause problems during training.

Relu Function:

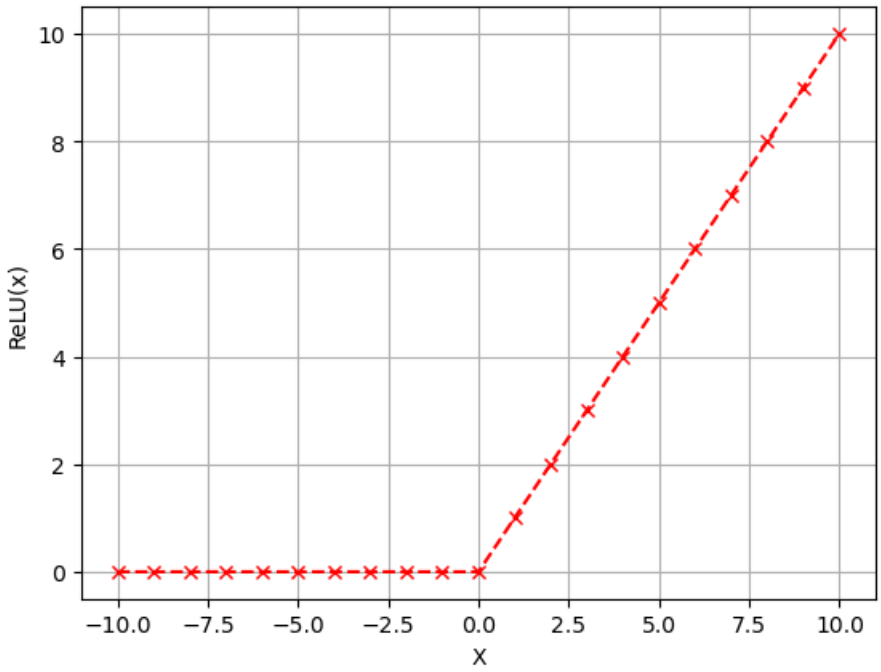
The Rectified Linear Unit (ReLU) is a popular activation function used in deep neural networks. It is a simple and effective function that computes the maximum between zero and the input. Mathematically, the ReLU function can be defined as:

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

where x is the input to the neuron.

The ReLU function is easy to compute and has been shown to be effective in improving the performance of neural networks. One of the advantages of the ReLU function is that it does not suffer from the vanishing gradient problem, which occurs when the gradient of the activation function approaches zero for very large or very small inputs. The ReLU function provides a constant gradient for all positive inputs, which can help speed up the training process. The ReLU function also introduces sparsity into the network by setting negative inputs to zero. This can help reduce the complexity of the network and prevent overfitting, which occurs when the model becomes too complex and fits the training data too closely.

```
y = np.maximum(0, x)
plot_graph(y, "ReLU(x)")
```



One potential drawback of the ReLU function is that it can lead to dead neurons, which are neurons that always output zero. This can occur if the neuron's weights are updated in such a way that it always outputs negative values. To address this issue, several variations of the ReLU function have been proposed, such as the Leaky ReLU and the Parametric ReLU.

Elu Function:

The Exponential Linear Unit (ELU) is a popular activation function used in deep neural networks. It is similar to the ReLU function, but it has some advantages over it. The ELU function computes the maximum between the input and a negative value transformed by an exponential function. Mathematically, the ELU function can be defined as:

f(x) = x, if x > 0

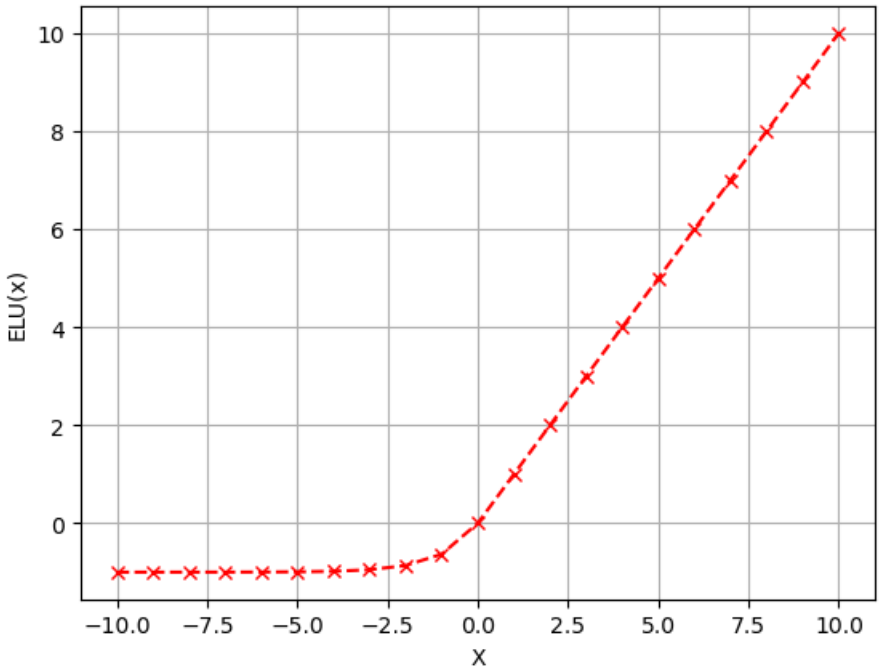
f(x) = alpha * (exp(x) - 1), if x <= 0

where x is the input to the neuron and alpha is a hyperparameter that controls the output when the input is negative.

```
alpha = 1.0

y = np.where(x>=0, x, alpha*(np.exp(x)-1))

plot_graph(y, "ELU(x)")
```



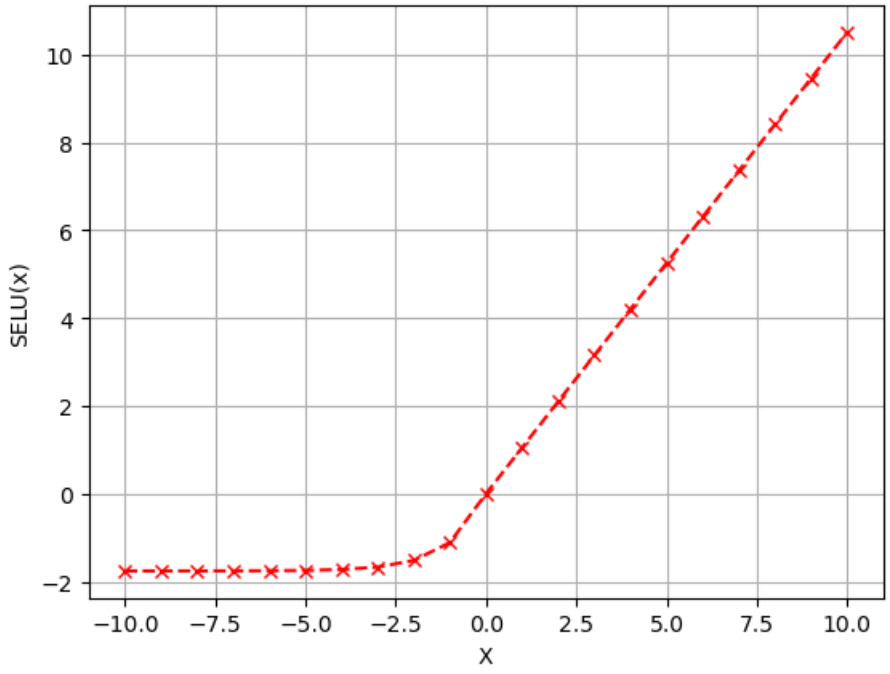
The ELU function has several advantages over the ReLU function. One of the advantages is that it reduces the bias shift, which occurs when the distribution of the activations becomes shifted towards positive values. This can improve the performance of the network and make it more robust to noise.

Another advantage of the ELU function is that it has a nonzero output when the input is negative. This helps to prevent dead neurons, which are neurons that always output zero. The nonzero output also helps to speed up the convergence of the network during training.

Selu Function:

The SELU (Scaled Exponential Linear Unit) activation function is a type of non-linear activation function used in deep learning neural networks. It is a variant of the ELU (Exponential Linear Unit) function and was introduced in 2017 by Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter in their paper "Self-Normalizing Neural Networks."

```
def selu(x, scale=1.0507, alpha=1.6733):
    return np.where(x > 0, scale*x, scale*(alpha*np.exp(x)-alpha))
plot_graph(selu(x), "SELU(x)")
```



The SELU function has several advantages over other activation functions. One of its most important features is that it is self-normalizing, meaning that the output of the function has zero mean and unit variance for any input with zero mean and unit variance, regardless of the depth of the network. This property helps to prevent the vanishing gradient problem, which can occur in deep neural networks when the gradients become too small to be useful for learning.

Moreover, the SELU function is smooth and continuous, which allows it to be used in gradient-based optimization algorithms for training neural networks. It is also a bounded function, which ensures that the output of the function remains within a certain range, preventing numerical instability during training.

Discussion & Conclusion:

In conclusion, the choice of activation function is an essential consideration when designing and training neural networks. Each function has its strengths and weaknesses, and the selection of an activation function depends on the specific requirements of the problem being solved. Researchers are continually exploring new activation functions to address the limitations of existing ones and to improve the performance of neural networks.