American International University-Bangladesh (AIUB)

Department of Computer Science Faculty of
Science &Technology (FST) Spring 22-23
COMPUTER VISION AND PATTERN RECOGNITION

Section: A Report
of Assignment 2 Date: 11
– March - 2023

Submitted By:

Serial No.	Student Name	Student ID
1.	Md Meraz Hossain	20-42460-1

Submitted to:
DR. DEBAJYOTI KARMAKER
Associate Professor, Faculty
Computer Science American International University-Bangladesh (AIUB)

Activation Function

An Artificial Neural Network needs activation functions to learn and understand extremely complex and non-linear functional mappings between the inputs and response/output variables. The network gains non-linear features as a result of them. Their primary function is to transform an artificial neural network node's input signal into an output signal. At the subsequent layer of the stack, that output signal is employed as an input. Transfer Function is another name for it.

Without the activation function, the change of the weights and bias would just be linear. Although a linear equation is straightforward to solve, it is not as powerful when it comes to solving complex issues or inferring complex functional mappings from data. Without an activation function, a neural network is merely a linear regression model..In general, linearity must be avoided by activation functions. Without them, the data would only move through the network's nodes and layers using linear functions (a*x+b). No matter how many layers the data is processed through, the output is always the result of a linear function because the composite of these linear functions is also a linear function.

There are numerous categories of activation functions, each with unique benefits and drawbacks.Sigmoid, tanh, ReLU, are a few of the most popular activation functions.

In this report, I am going to discuss **Step, Sigmoid, Tanh, ReLU, Elu, and Selu**.

Step Activation Function

The perceptron network employs the Step activation function. This is typically employed in single-layer networks to provide binary (0 or 1) or bipolar output (-1 or 1). These are known respectively as Binary Step Function and Bipolar Step Function. Here, a function's output is 1 (a neuron will fire) if the input value exceeds a threshold value; otherwise, it is 0 or -1 in the case of a bipolar step function (neuron will not fire).When an input value crosses a threshold, the step function's output quickly changes from 0 to 1 and is discontinuous. The step function is less suitable for training neural networks using gradient-based optimization methods since the derivative is usually always 0. As a result, updating the network's weights during backpropagation is difficult.

The step function has the following mathematical definition:

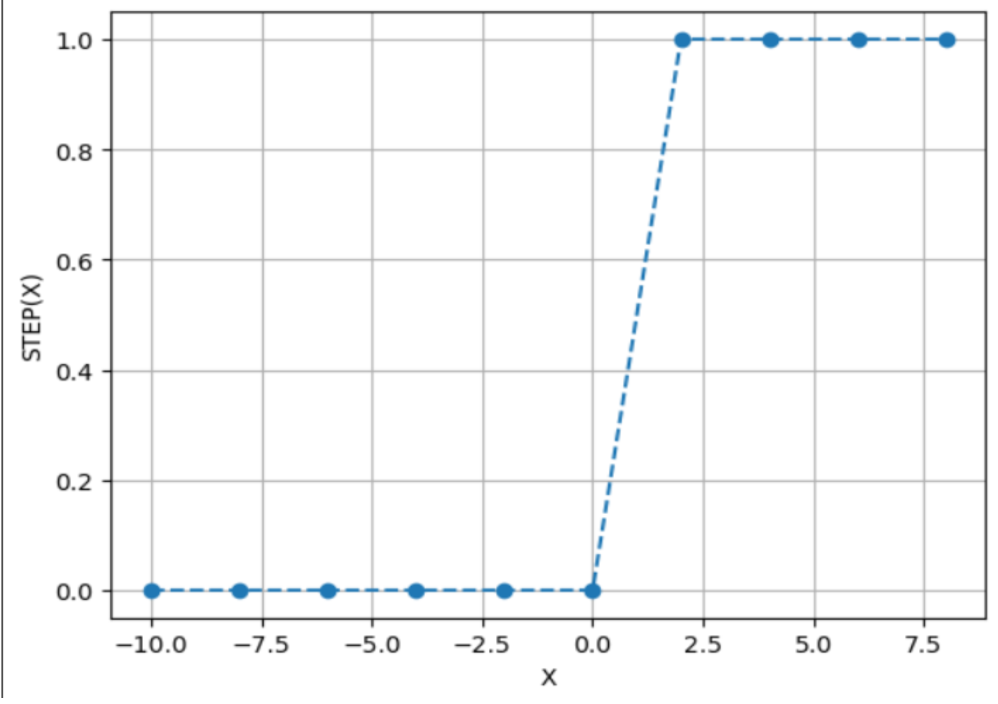
$f(x) = 0$, if x 0,

f(x) = 1, if x >= 0.

Step activation function is defined as:

```
y = list(map(lambda n: 1 if n>0.5 else 0, x))
plot_graph(y,"STEP(X)")
```

Resulted Graph:



Advantages:

For some binary classification issues, the step activation function is an easy and effective approach to generate binary output. As its result is a distinct decision boundary, it is simple to comprehend and interpret.

Disadvantages:

The step function has a zero gradient. Since the gradients of the activation functions are supplied for error computations during back-propagation in order to refine and optimize the outputs, this renders the step function less useful. It cannot be used for multi-class classification.

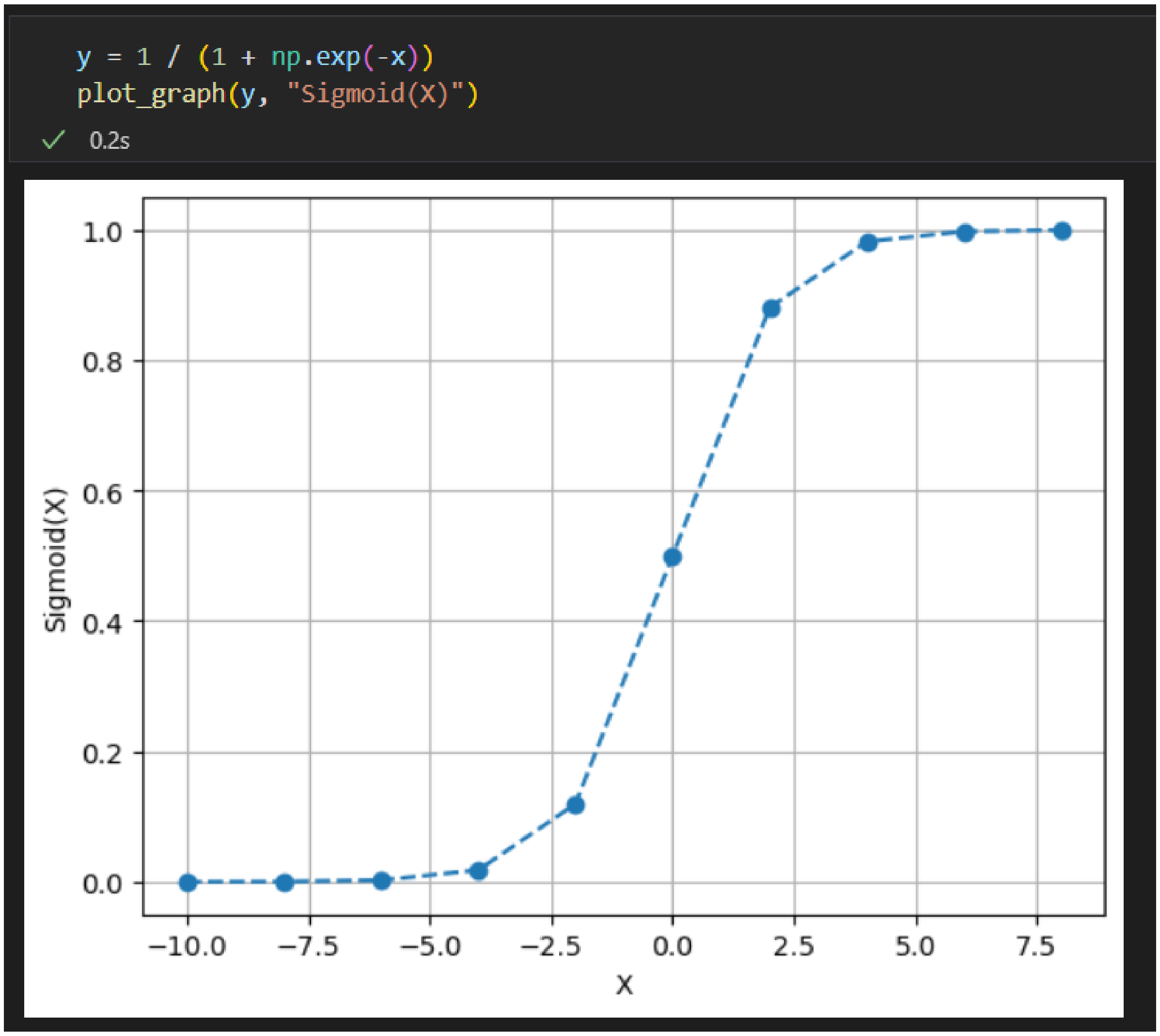
Sigmoid Activation Function

As an activation function, the sigmoid function is a class of mathematical functions that is frequently utilized in logistic regression models and neural networks. Any input value is converted into an output value between 0 and 1. The sigmoid function is used to transform the outcome after computing the weighted sum of the inputs. The sigmoid function's output can be understood as the likelihood that the input belongs to a particular category.

The sigmoid activation function is defined by the mathematical expression:

σ(z) = 1 / (1 + e^(-z)),

Sigmoid activation function is defined as:



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

✓ 0.2s

Advantages:

Every point can be derived from it. Any activation function would like to have this quality. This means that we can determine the sigmoid curve's gradient at any two points, which will aid in the backpropagation of model error. Backpropagation becomes difficult if an activation function cannot be derived at each point because gradients cannot be obtained for those sites. The output values are limited to a range of 0 to 1. This guarantees that the activations won't blow up. At times, this function provides precise forecasts. This indicates that the output is brought close to 1 or 0 for input values more than 2 or less than -2, respectively.

Disadvantages:

The issue of diminishing gradient is one of the main drawbacks of utilizing sigmoid. As you can see in the illustration, the sigmoid's derivative is very low for extremely high or extremely low values of x. The derivative's maximum value is 0.25. If you recall, when calculating gradients for backpropagation, we apply the chain rule. Chain rule causes the number of factors in the product to keep growing as we approach the input layer. These variables are few in number. As we get closer to the beginning, the product continues to decline. The network may then refuse to learn more or become too slow to make a reliable forecast as a result. It saturates and kills gradients. Refer to the figure of the derivative of the sigmoid. At both positive and negative ends, the value of the gradient saturates at 0. That means for those values, the gradient will be 0 or close to 0, which simply means no learning in backpropagation.

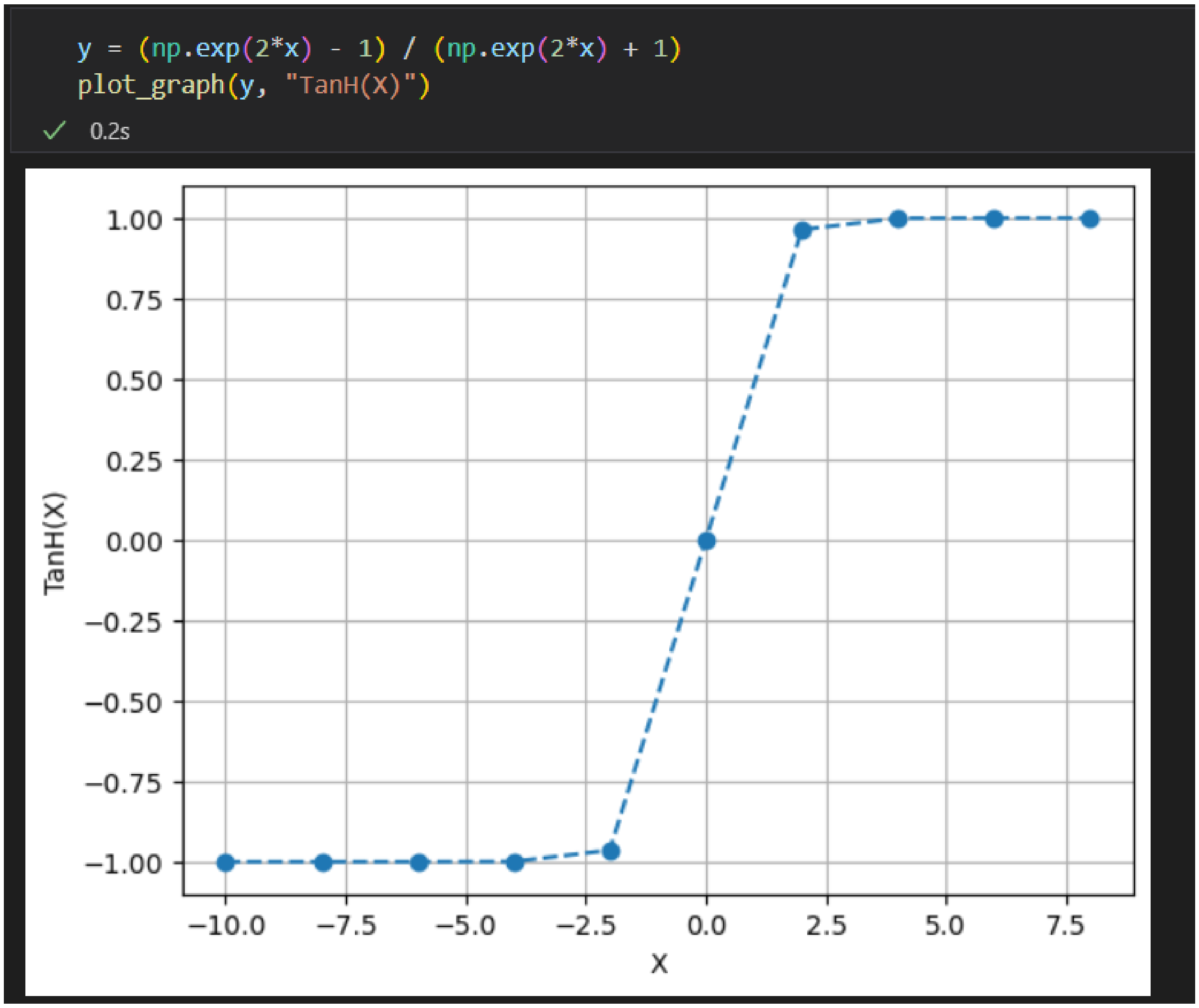
Tanh Activation Function

Tanh, commonly known as a hyperbolic tangent, is a well-liked activation function in neural networks. It is a mathematical formula that converts inputted numbers into numbers between (-1) and (1) range. Tanh transforms the value you supply it with into a number between -1 and 1. The function, in more precise terms, multiplies the exponential function of the input by the product of the exponential functions of the input and the negative input, subtracts the exponential function of the negative input, and divides the outcome by the exponential function of the input.

The mathematical expression of the Tanh activation function is:

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

Tanh activation function is defined as:



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

✓ 0.2s

Advantage:

Similar benefits to sigmoid exist, but because it is zero-centered, it may be preferable. Tanh produces an output between -1 and 1. Thus, a problem with the sigmoid activation function is resolved. It is non-linear and has the ability to model complex relationships between input and output variables. Tanh is advantageous in backpropagation and other gradient-based optimization techniques.

Disadvantage:

It also has the problem of vanishing gradient but the derivatives are steeper than that of the sigmoid. Hence making the gradients stronger for tanh than sigmoid. As it is almost similar to sigmoid, tanh is also computationally expensive. Similar to sigmoid, here also the gradients saturate.

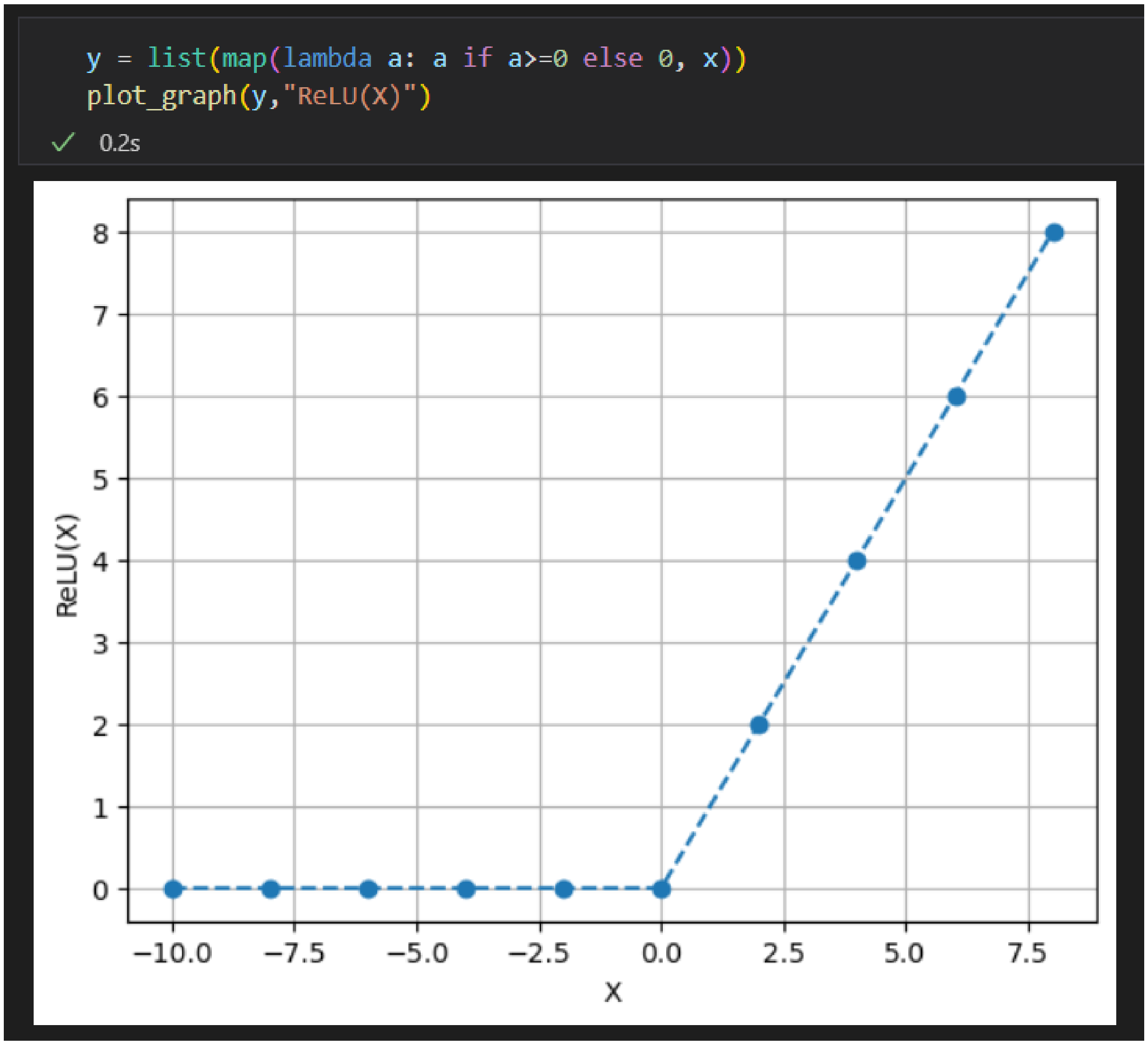
Relu Activation Function

The ReLU (Rectified Linear Unit) function is a popular activation function in neural networks and deep learning. The ReLU function is a simple piecewise linear function that converts all input values that are less than or equal to zero to zero and all input values that are greater than zero to the same value.

The ReLU function has the following mathematical definition:

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

ReLU activation function is defined as:



```
y = list(map(lambda a: a if a>=0 else 0, x))
plot_graph(y,"ReLU(X)")
```

✓ 0.2s

Advantage

Due to the ReLU function's non-linear nature, it is simple to backpropagate errors and have many layers of neurons active. Compared to the sigmoid and tanh functions, it was discovered to significantly speed up the convergence of stochastic gradient descent.

Not all of the neurons are activated simultaneously. Only a few neurons are triggered since some neurons have zero output, making the network sparse, effective, and simple to compute.

Disadvantage

At zero, it is not differentiable, and ReLU is unbounded. Since the gradients for negative input are 0, backpropagation does not update the weights for activations in that region. As a result, dead neurons may develop that never fire. By lowering the bias and learning rate, this can be managed. The performance of the neural network is affected by ReLU output since it is not zero-centered. Backpropagation weights have a gradient that is either entirely positive or entirely negative. This could cause the gradient updates for the weights to have unwanted zig-zagging behavior. Batchnorm is capable of handling this. This problem is slightly mitigated by batchnorm, which adds up these gradients over a batch of data such that the final update for the weights can have various signs.

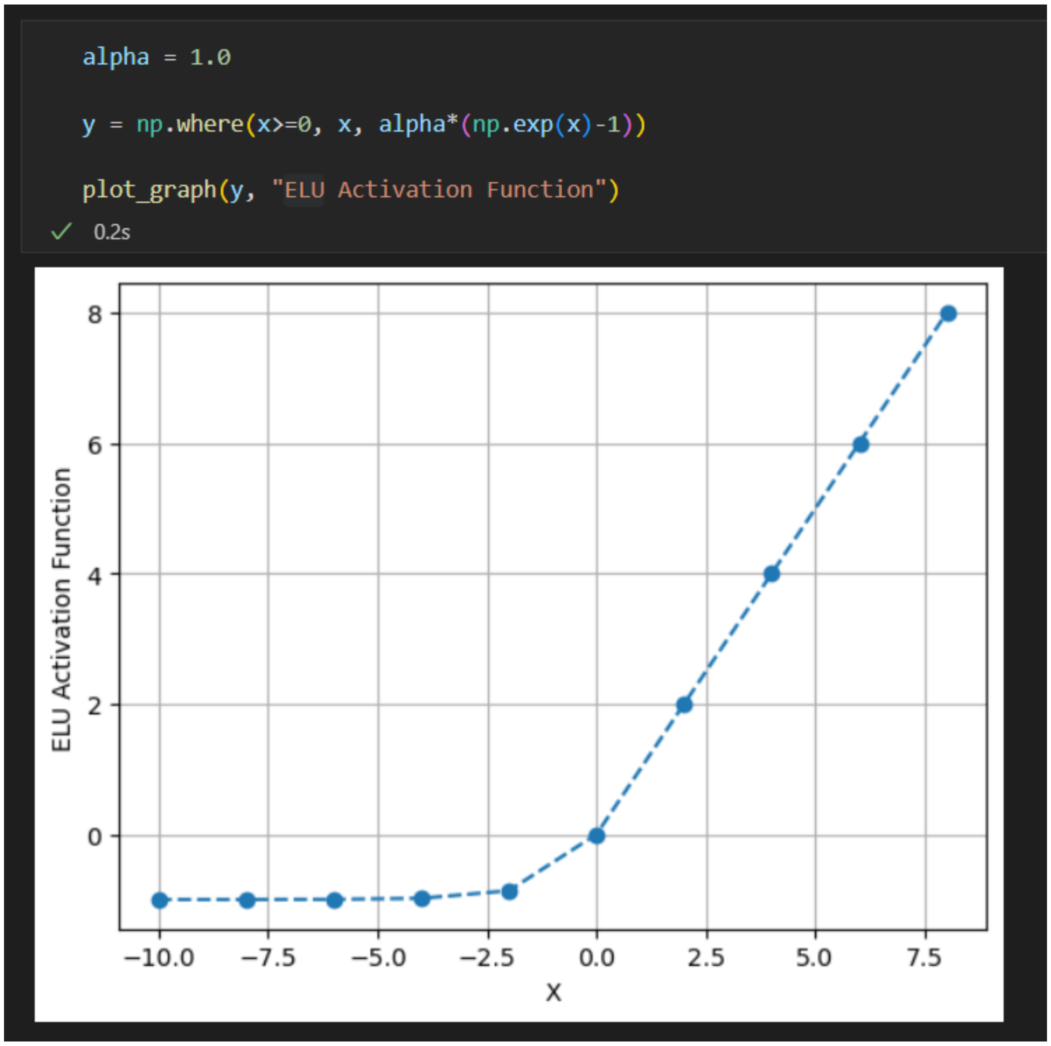
Elu Activation Function

Exponential Linear Unit, or ELU, is a type of activation function utilized in neural networks. An effective method for creating neural networks that can learn intricate correlations between inputs and outputs is the ELU activation function. It is a prominent option in deep learning due to its capacity to reduce the vanishing gradient problem, enhance performance, and resistance to noise. The activation of the ELU

The ELU function has the following mathematical definition

f(x) = x, x >= 0 f(x) = alpha * (e^x - 1), x < 0

The ELU function is defined as:



```
alpha = 1.0

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

plot_graph(y, "ELU Activation Function")
```

✓ 0.2s

Advantage :

The ELU function, unlike the ReLU function, can help prevent "dead" neurons while having a nonzero output for negative inputs. Furthermore smooth and continuously differentiable, the ELU function may in some cases make optimization easier.

Disadvantage :

It has a problem with dying ReLUs. Negative values will always be discarded by ReLU . As a result, the gradient of these units will also become 0, and as we are well aware, a gradient of 0 indicates that there will be no weight updating during backpropagation. Simply explained, the neurons that enter this state will stop responding to input errors or deviations. As a result, the model is less able to accurately fit the data.

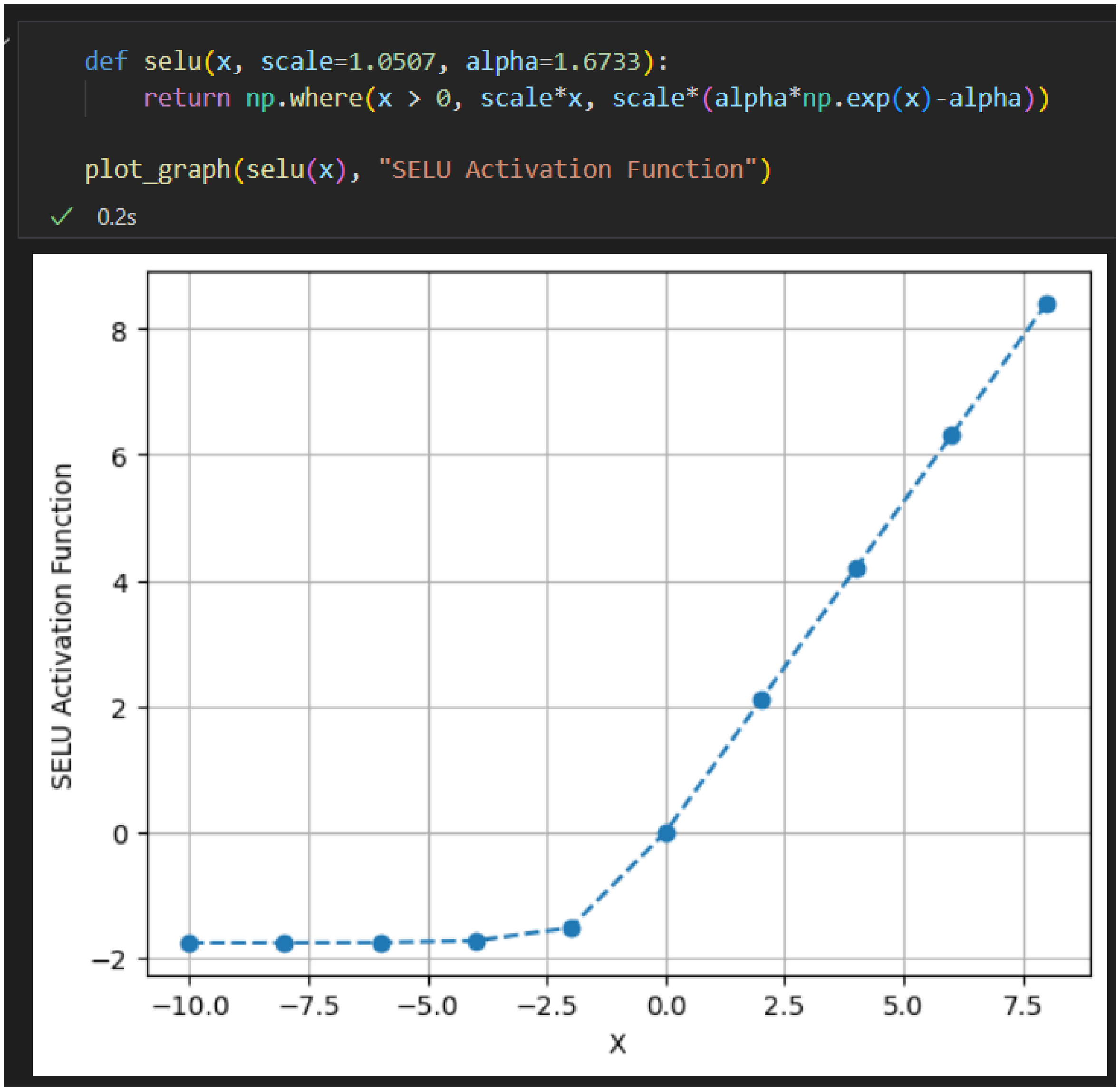
Selu Activation Function

SELUs, or Scaled Exponential Linear Units, are activation functions that induce self-normalization. SELU network neuronal activations automatically converge to a zero mean and unit variance.If x is larger than 0, the output result is x multiplied by lambda. If the input value x is less than or equal to zero, we have a function that goes up to 0, which is our output y, when x is zero. Essentially, when x is smaller than zero, we take the exponential of the x-value minus 1, then we multiply it with alpha α and lambda λ.

The SELU function has the following mathematical definition

f(x) = lambda * (max(0, x) + alpha * (exp(min(0, x)) - 1))

The SELU function is defined as:



```
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 Activation Function")
```

✓ 0.2s

Advantage:

The main advantage of the SELU function is that it can improve the performance and stability of deep neural networks, particularly when the input is highly dimensional or complexly structured. The self-normalizing property of the SELU function mitigates the deep network vanishing and ballooning gradient problems that may cause poor performance or unstable training.

Disadvantage:

There are several limitations on the SELU function, though. It requires proper initialization of the network weight and might not function as well with specific data types or architectural settings. Also, it uses more processing power compared to some other activation mechanisms like ReLU. Overall, the SELU function is a potential choice for deep neural networks that need to be stable and perform well, although it might not be appropriate for all applications.