AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

COMPUTER VISION AND PATTERN RECOGNITION [C][SPRING 22-23]

Report on Activation Function

Submitted To:

DR. DEBAJYOTI KARMAKER

Associate Professor, Computer Science

Submitted By:

Name: TAZIN ALAM

ID: 20-43848-2

Department of CSE, AIUB

Abstract:

"The world is one big data problem." — as it turns out this saying holds true both for our brains as well as machine learning. Every single moment our brain is trying to segregate the incoming information into the "useful" and "not-so-useful" categories. A similar process occurs in artificial neural network architectures in deep learning. The segregation plays a key role in helping a neural network properly function, ensuring that it learns from the useful information rather than get stuck analyzing the not-useful part. And this is also where activation functions come into the picture. [1]

Keyword: Neural Network, Activation function, advantages, Sigmoid, Tanh, Relu, Elu, Selu.

Introduction:

Activation functions are a critical component of neural networks as they introduce non-linearity and help in the mapping of inputs to outputs in a non-linear manner. An Activation Function decides whether a neuron should be activated or not. Consequently, it will determine using more basic mathematical operations whether the neuron's input to the network is significant or not when making a prediction.

Types of Activation Function:

There are several activation functions that are commonly used in deep learning models. 3 most popular types of Activation Functions in Neural Network are – (i) Binary Step Function (ii) Linear Activation Function & (iii) Non-Linear Activation Functions

In this report, binary step function and some non-linear activation function (such as Sigmoid, Tanh, Relu, Elu, Selu)will be described with their advantages and disadvantages.

Binary Step Function:

The simplest type of activation function used in neural networks. With a scalar input and a binary output, it is a threshold-based function. Depending on whether the input is greater than or less than a predetermined threshold value, the output is either 0 or 1.

Mathematically, the binary step function can be defined as:

f(x) = { 1, if x >= 0; 0, otherwise }

Advantage:

-Simple and easy to implement

Disadvantages:

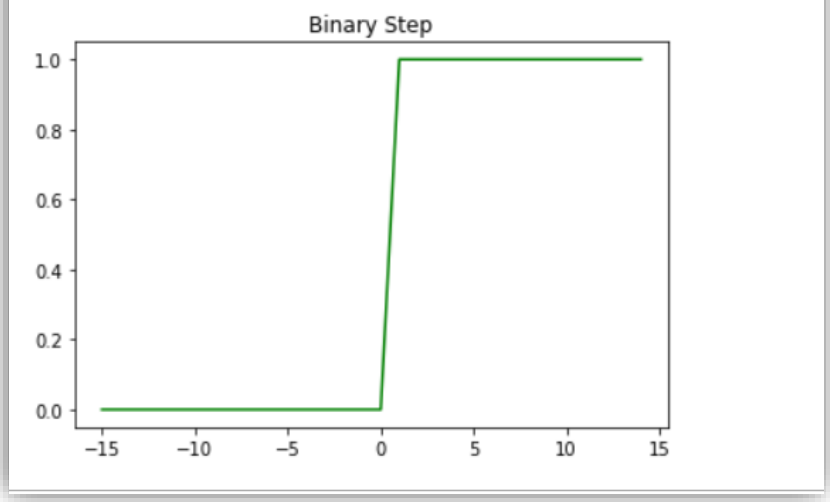
- Does not allow multi-value outputs

- Not differentiable

- Unsuitable for use in backpropagation-based learning algorithms

The function can be visualized as a step-like graph, where the output jumps from 0 to 1 when the input crosses the threshold value.

```python
import matplotlib.pyplot as plt
import numpy as np
#binary activation function y=f(x)==>y=x if x<=0
x=np.arange(-15,15)
y=[]
for i in range(x.shape[0]):
    if(x[i]>0):
        y.append(1)
    else:
        y.append(0)
plt.plot(x,y,c="green")
plt.title("Binary Step")
plt.show()
```



Sigmoid Function:

The activation function, which is frequently used, converts any input value to a range between zero and one. It has a smooth derivative that is appropriate for training backpropagation. It can, however, experience vanishing gradients, which makes it challenging to train deep neural networks.

Advantages:

-To avoid "jumps" in the output values, use a smooth gradient.

- 0–1 output values are used to normalize each neuron's output.

Disadvantages:

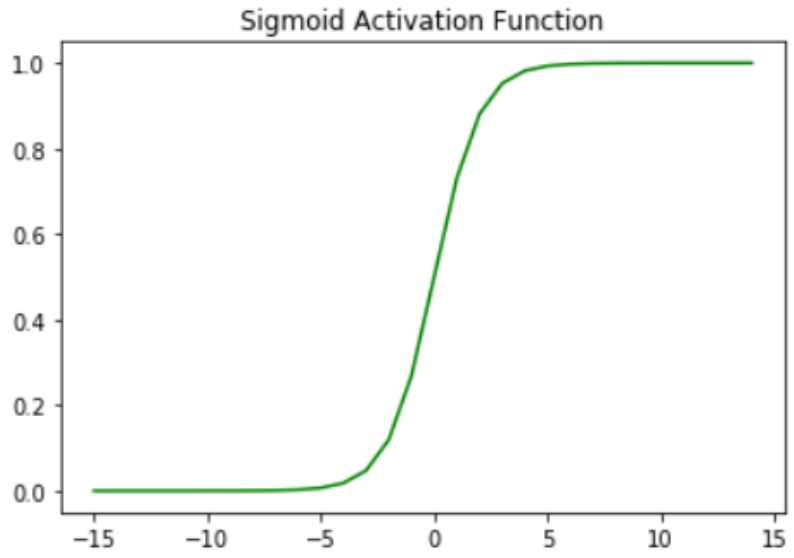
- A vanishing gradient problem results from the almost complete lack of change in the prediction for extremely high or extremely low values of X.

- Outputs not zero centered.

-Computationally expensive

The graph would be like-

```python
import matplotlib.pyplot as plt
import numpy as np
#Sigmoid Activation function y=1/(1+e^-x)
x=np.arange(-15,15)
y=[]
for i in range(x.shape[0]):
    y.append(1/(1+np.exp(-x[i])))
plt.plot(x,y,c="green")
plt.title("Sigmoid Activation Function")
plt.show()
```



TanH Function(Hyperbolic Tangent):

The sigmoid function is comparable, but this function maps any input value to a range between -1 and 1. It has a smooth derivative that is appropriate for training backpropagation.

Advantages:

Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.

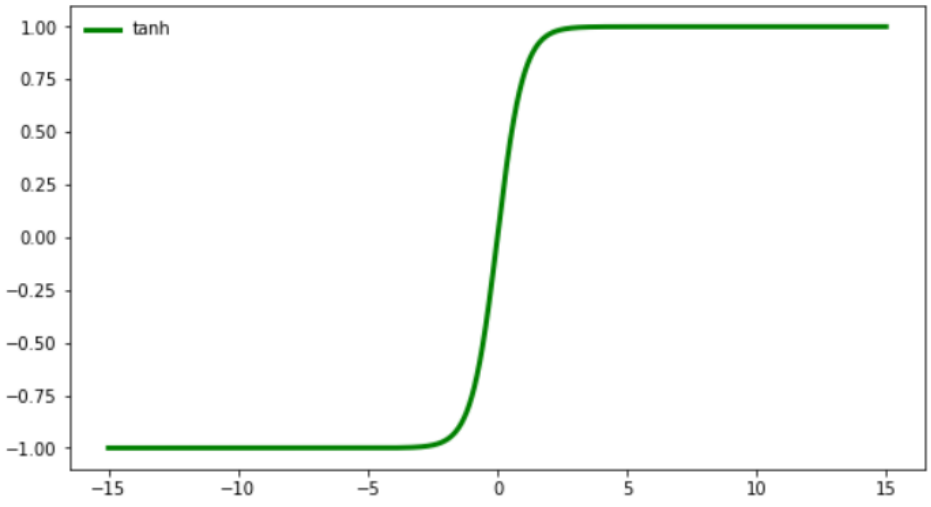
Disadvantages:

- vanishing gradient

The graph would be like-

```python
import matplotlib.pyplot as plt
import numpy as np

def tanh(x):
    t=(np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x))
    return t
```

```python
x=np.arange(-15,15,0.01)
plt.ax=plt.subplots(figsize=(9,5))
plt.plot(x,tanh(x), color="green", linewidth=3, label="tanh")
plt.legend(loc="upper left", frameon=False)
plt.show()
```



Relu Function (Rectified Linear Unit):

It is a popular activation function that maps any input below zero to zero and any input above zero to itself. It is simple, computationally efficient, and has no vanishing gradient problem.

Advantages

-Computationally efficient

-ReLU has a derivative function and allows for backpropagation

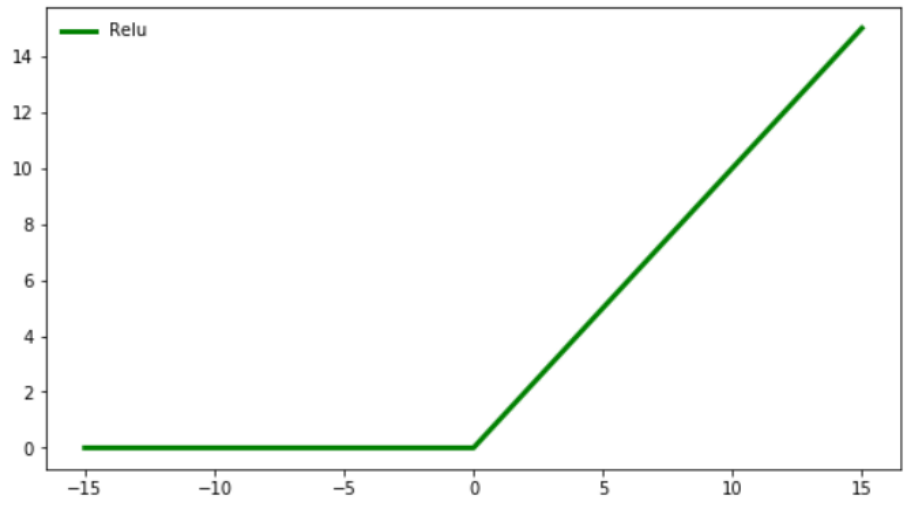
Disadvantages

-The Dying ReLU problem, When inputs are close to zero or negative, the function's gradient becomes zero, making it impossible for the network to use backpropagation and learn

The graph is –

```python
import matplotlib.pyplot as plt
import numpy as np

def relu(x):
    y=max(0,x)
    return y
```

```python
x=np.arange(-15,15,0.01)
relu_output=[relu(i) for i in x]
plt.ax=plt.subplots(figsize=(9,5))
plt.plot(x,relu_output, color="green", linewidth=3, label="Relu")
plt.legend(loc="upper left", frameon=False)
plt.show()
```



Elu (Exponential Linear Unit):

ELU is very similar to RELU except negative inputs. They are both in identity function form for non-negative inputs. On the other hand, ELU becomes smooth slowly until its output equal to −α−  whereas RELU sharply smoothest.

Advantages:

-ELU is not piece-wise linear, this makes it model the non-linearity better.

-Helps alleviate the "dying ReLU" problem and can achieve higher accuracy than the ReLU function.

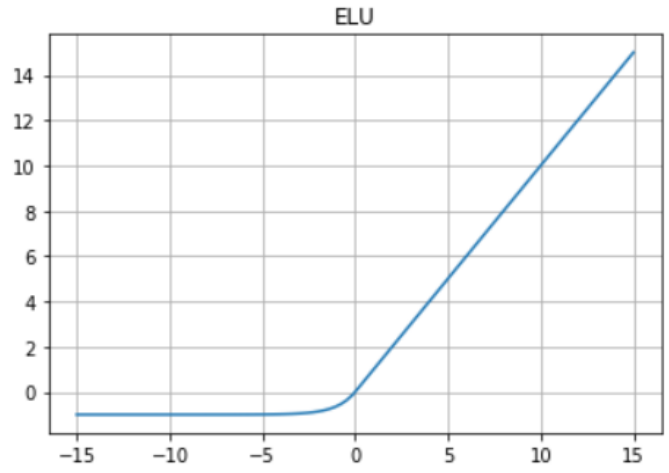
Disadvantages:

-alpha is fixed, not learned.

- It still suffers from Gradient Exploding and Gradient Vanishing Problem.

-Computationally more expensive than the ReLU function.

```python
import matplotlib.pyplot as plt
import numpy as np

def elu(x, alpha=1.0):
    return np.where(x > 0, x, alpha * (np.exp(x) - 1))

# for input
x = np.linspace(-15, 15,100)

# output values
y = elu(x)


plt.plot(x, y)
plt.title("ELU")
plt.xlabel("")
plt.ylabel("")
plt.grid()
plt.show()
```



Selu(Scaled Exponential Linear Units):

It is a self-normalizing variant of the ReLU function, which can achieve better accuracy than other activation functions. It has a mean activation of zero and a standard deviation of one, which helps speed up convergence during training.

Advantages:

-does not have vanishing gradient problem

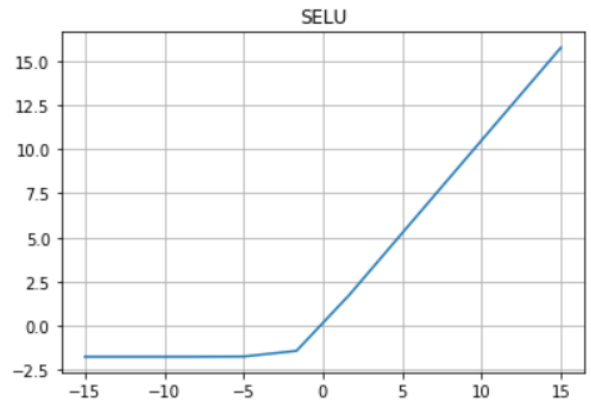
-Learn faster and better than other activation functions without needing further procession

Disadvantages:

-As a relatively new activation function so it is not yet used widely in practice

-Requires specific initialization and is computationally more expensive than the ReLU function.

```python
import matplotlib.pyplot as plt
import numpy as np

def selu(x, alpha=1.67326, scale=1.0507):

    return scale * np.where(x > 0, x, alpha * (np.exp(x) - 1))

# input
x = np.linspace(-15, 15, 10)

# output
y = selu(x)

# Plot the SELU activation function
plt.plot(x, y)
plt.title("SELU")
plt.xlabel("")
plt.ylabel("")
plt.grid()
plt.show()
```



A difference table has been given here which was collected from online platform

Activation Function	Range of Output	Nonlinear	Monotonic	Smooth	Differentiable	Advantages	Disadvantages
Binary Step	0 or 1	No	No	No	No	Simple, easy to implement	Not differentiable, cannot be used in backpropagation
Sigmoid	0 to 1	Yes	Yes	Yes	Yes	Smooth, differentiable	Susceptible to vanishing gradient problem
Tanh	-1 to 1	Yes	Yes	Yes	Yes	Centered around zero, smooth, differentiable	Susceptible to vanishing gradient problem
ReLU	0 to infinity	Yes	No	No	No	Fast computation, avoids vanishing gradient problem	Output can be non-positive, susceptible to dying ReLU problem

Activation Function	Range of Output	Nonlinear	Monotonic	Smooth	Differentiable	Advantages	Disadvantages
Leaky ReLU	0 to infinity	Yes	No	No	Yes	Overcomes dying ReLU problem	Output can be non-positive
ELU	-1 to infinity	Yes	Yes	Yes	Yes	Reduces bias shift, overcomes dying ReLU problem	Computationally expensive
SeLU	Real numbers	Yes	Yes	Yes	Yes	Self-normalizing, overcomes dying ReLU problem	Computationally expensive, only works with specific hyperparameters

Discussion and Conclusion:

In summary, choosing the appropriate activation function depends on the task at hand, the size and complexity of the network, and the data distribution. The ELU and SELU functions can attain higher precision but are computationally more expensive, making the ReLU function the most often employed activation function.

In conclusion, activation functions are a crucial part of neural networks because they enable the addition of nonlinearity to the model. The binary step function, sigmoid, tanh, relu, elu, and selu are only a few of the many activation functions that have been created and employed; each has pros and cons. It is crucial to select the right activation function for a given issue because it might have a big impact on the model's performance. To make a wise choice, researchers and practitioners must be aware of the properties of each activation function.

References:

[1]

https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=drive%20V7's%20tools.-,What%20is%20a%20Neural%20Network%20Activation%20Function%3F,prediction%20using%20simpler%20mathematical%20operations.

https://indiantechwarrior.com/7-types-of-neural-network-activation-functions-how-to-choose/