RA2311047010044 38

INDEX

NAME K.D.D.SaiAbhiram SUBJECT Deep learning lab obsb

STD. _____ DIV. _____ ROLL NO. _____ SCHOOL _____

SR. NO.	PAGE NO.	TITLE	DATE	TEACHER'S SIGN / REMARKS
1		Analyze and exploring the deep learning platforms	24-07-25	
2		Implement a classifier using open-source data set	7/8/25	
3.		Study of the classifiers with respect to statistical parameters	7/8/25	
4.		Build a simple feed forward neural network to recognize hand written characters	14/8/25	
5.		Study of activation function and its role	28-8-25	
6.		Implement gradient descent & back propagation in deep neural network.	19-9-25	

22/08/25

5. Study of activation function and its role

Aim:- to study common neural network activation functions sigmoid, tanh, relu, leaky relu & softmax and analyze their roles.

description (objective)

① Sigmoid :- $\sigma(z) = \frac{1}{1+e^{-z}}$

maps to (0,1) smooth, Probabilistic interpretation. (Pros)
Downside : large |z| (∞) [0, 1]

② $\text{Tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Range (-1,1) sigmoid can suffer vanishing gradients (Cons)

③ Relu (rectified linear unit) : $\text{Relu}(z) = \max(0, z)$

$z > 0$; risk of dead relus when neuron's keep outputting 0 $[0, \infty]$

④ Leaky relu :- $\text{L Relu}(z) = \max(\alpha z, z)$; $\alpha > 0$ (eg :- 0.01)

$(-\infty, \infty)$ It accepts small (α) allows negative slope helps to reduce dead neurons

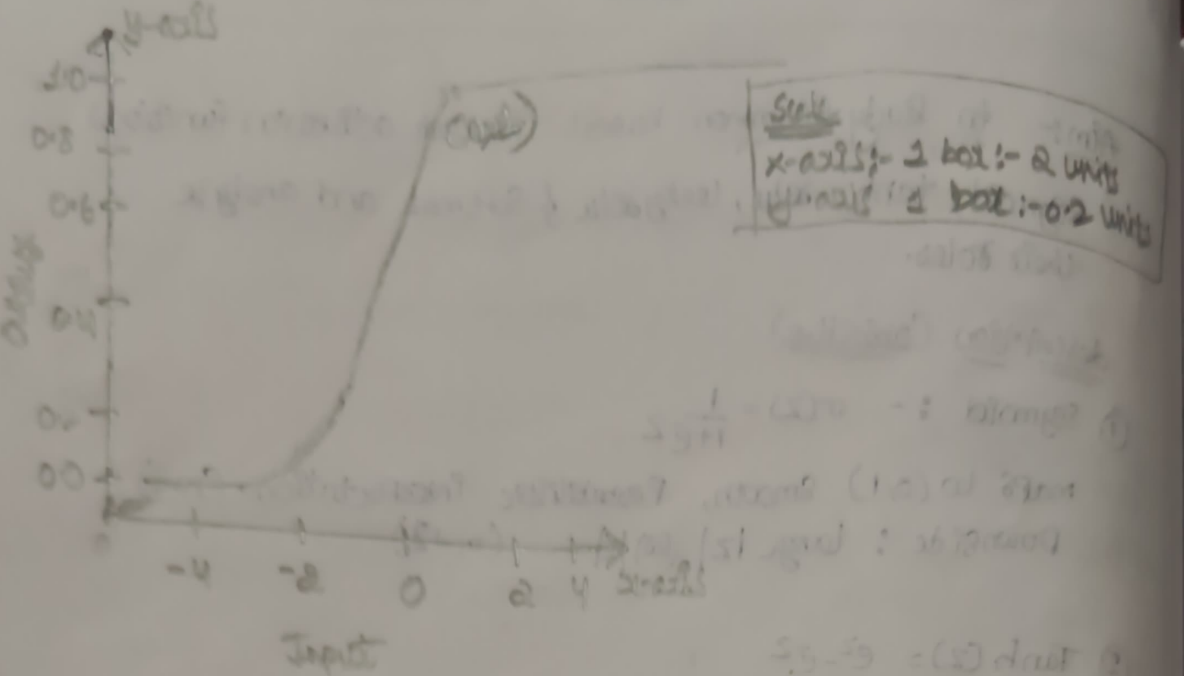
⑤ Softmax :- $P_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$ Cross-entropy loss for multi class classification

(0, 1)

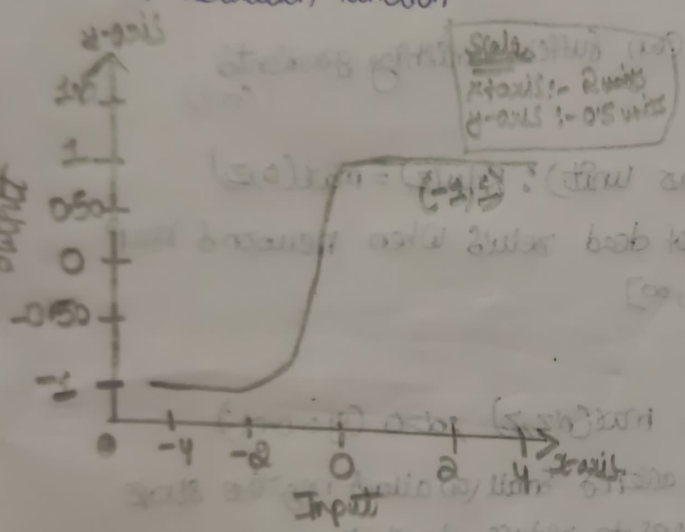
Pseudo code :-

1. Start
2. Import necessary libraries
3. Define activation function.
   - (i) Sigmoid (x) = 1/(1+e^(-x))
   - (ii) Tanh (x) = (e^x - e^(-x))/(e^x + e^(-x))
   - (iii) Relu (x) = (max (0, x)
   - (iv) Leaky relu = max (0)(z, z)
4. Generate Input values in a range
5. Apply each activation function on the Input value
6. Plot the output of each function to visualise their behaviour
7. Compare and observe range outputs
   - non-linearity
   - suitable for classification and regression.

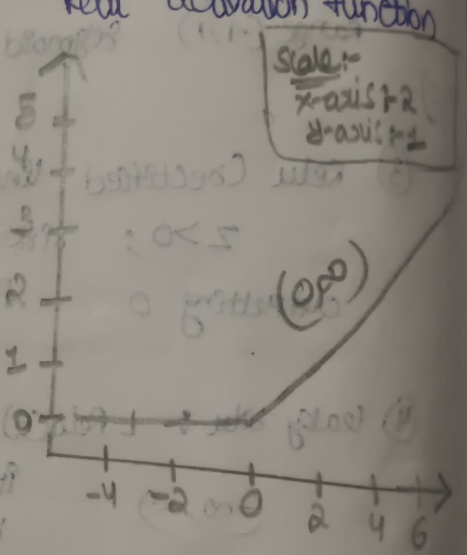
Sigmoid Activation function



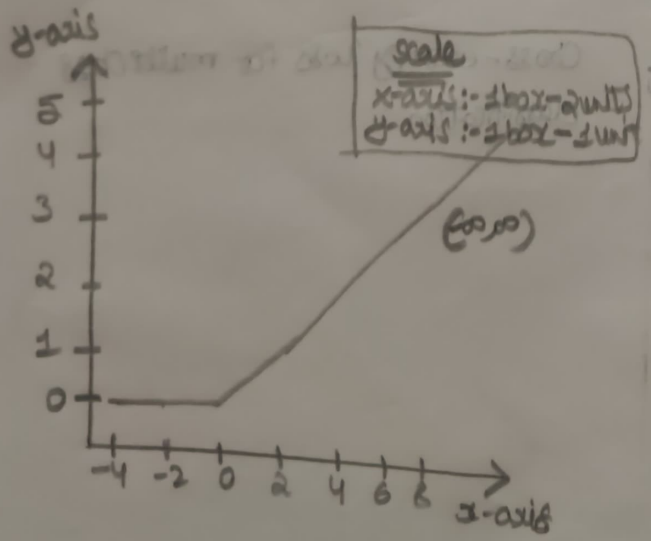
Tanh activation function



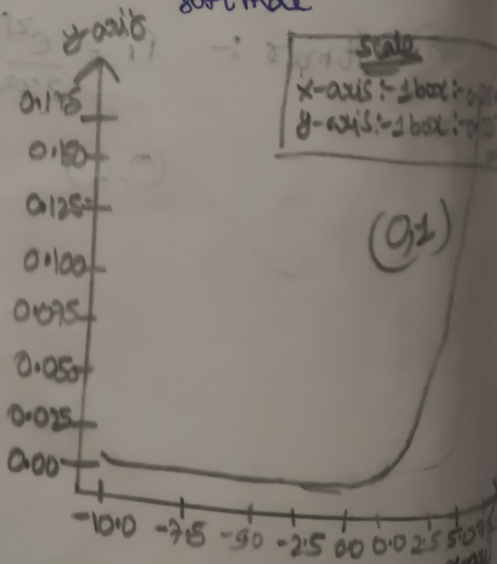
Relu activation function

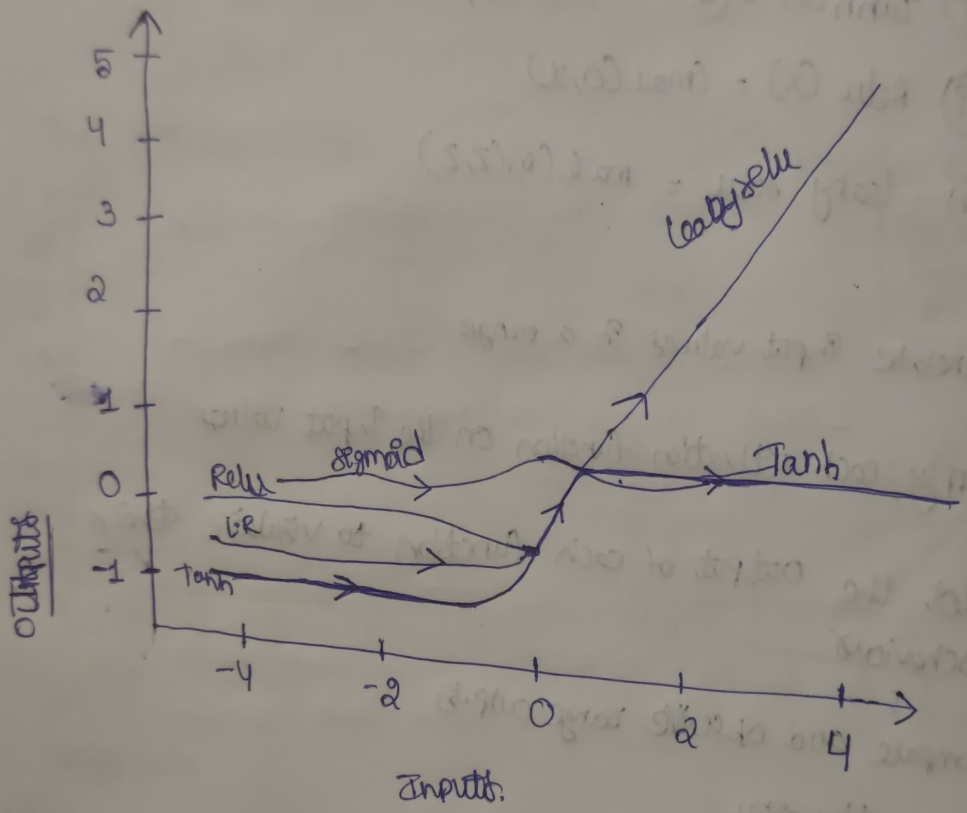


Leaky relu



Soft max





Sigmoid : 0 to 1
tanh : -1 to 1
Relu : -ve to 0
Leaky relu

Observation

① Sigmoid :- outputs are compressed between 0 and 1 but prone to vanishing gradients.

② Relu :- outputs are zero from negative inputs.

③ Tanh :- outputs range from -1 to 1 but suffers and vanishing gradients for large inputs

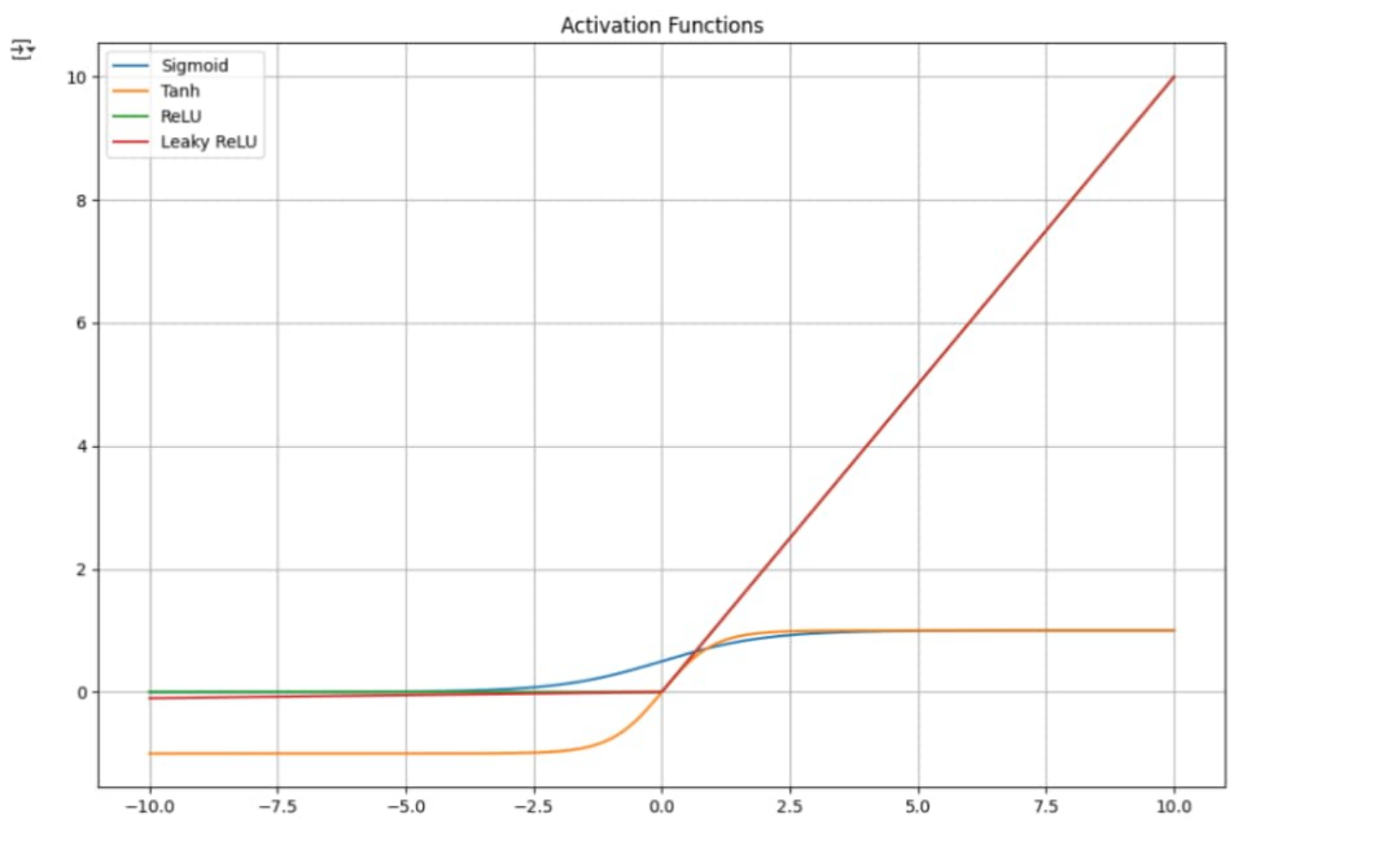
④ Leaky Relu :- Allows small negative outputs

Results :- Successfully implemented activation functions and diff role.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LeakyReLU, Activation

# Plot activation functions
def plot_activation_functions():
    x = np.linspace(-10, 10, 400)

    # Sigmoid
    sigmoid = 1 / (1 + np.exp(-x))

    # Tanh
    tanh = np.tanh(x)

    # ReLU
    relu = np.maximum(0, x)

    # Leaky ReLU
    alpha = 0.01
    leaky_relu = np.where(x > 0, x, alpha * x)

    # Plot all
    plt.figure(figsize=(12, 8))
    plt.plot(x, sigmoid, label='Sigmoid')
    plt.plot(x, tanh, label='Tanh')
    plt.plot(x, relu, label='ReLU')
    plt.plot(x, leaky_relu, label='Leaky ReLU')
    plt.title("Activation Functions")
```

```
# Load and prepare data
data = load_iris()
X = data.data
y = data.target.reshape(-1, 1)

encoder = OneHotEncoder(sparse_output=False)
y_encoded = encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define a function to build, train, and evaluate model with a given activation function
def train_with_activation(activation_name):
    print(f"\nTraining with activation function: {activation_name}")
    model = Sequential()

    if activation_name == 'leaky_relu':
        # For LeakyReLU, add layer without activation then LeakyReLU layer
        model.add(Dense(10, input_shape=(4,)))
        model.add(LeakyReLU(alpha=0.01))
        model.add(Dense(10))
        model.add(LeakyReLU(alpha=0.01))
```

```
        model.add(Dense(10, activation=activation_name))

    # Output layer always softmax for multi-class classification
    model.add(Dense(3, activation='softmax'))

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    model.fit(X_train, y_train, epochs=50, batch_size=5, verbose=0)
    loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
    print(f"Test accuracy with {activation_name}: {accuracy:.4f}")

# Plot activation functions first
plot_activation_functions()

# Train and evaluate models with different activation functions
for act_func in ['sigmoid', 'tanh', 'relu', 'leaky_relu']:
    train_with_activation(act_func)
```


Activation Functions
Sigmoid
Tanh
ReLU
Leaky ReLU
10
8
6
4
2
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

```
Training with activation function: sigmoid
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models,
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Test accuracy with sigmoid: 0.9000

Training with activation function: tanh
Test accuracy with tanh: 1.0000

Training with activation function: relu
Test accuracy with relu: 1.0000

Training with activation function: leaky_relu
/usr/local/lib/python3.12/dist-packages/keras/src/layers/activations/leaky_relu.py:41: UserWarning: Argument `alpha` is deprecated. Use `negative_slope` instead.
  warnings.warn(
Test accuracy with leaky_relu: 1.0000
```