EXP NO: 5 :- Study of Activation functions and its role.

Aim :-

To study the different activation functions used in neural networks and analyze their role in introducing non-linearity, enabling the network to learn complex patterns.

Objectives :-

1. To understand the mathematical behavior of commonly used activation functions.
2. To implement Sigmoid, Tanh, ReLu, and softmax functions.
3. To Compare the effect of different activation functions on model performance
4. To observe the importance of non-linearity in neural network.

Algorithm :-

1. Define activation functions: Sigmoid, Tanh, ReLU, softmax
2. Build a simple neural network (Feed Forward) for classification (MNIST dataset).
3. Train multiple models with different activation functions
4. Record accuracy and learning behavior
5. Compare results to analyze which activation functions performs best.

Pseudo Code:-

START

Import necessary libraries

Define activation functions:

Sigmoid(x) = 1/(1+exp(-x))

Tanh(x) = (exp(x) - exp(-x))/(exp(x) + exp(-x))

ReLU(x) = max(0, x)

softmax(x) = exp(x)/sum(exp(x))

Load dataset. (MNIST)

For each activation function in [sigmoid, Tanh, ReLU]:

Build a neural network with that activation

Train the network on training data

Evaluate on test data

store accuracy

Compare accuracy results

END

Observations:-

1. Sigmoid:- Smooth, but suffers from vanishing gradient for large positive/negative inputs.
2. Tanh:- Better than sigmoid, outputs range (-1, 1), but still faces vanishing gradient.
3. ReLU:- Most effective in deep networks, avoids vanishing gradient, fast Convergence

4. Softmax:- Used in output layer for multi-class classification problems

5. Experiments show ReLU achieves faster convergence and higher accuracy compared to sigmoid

Conclusion:-

Activation functions introduce non-linearity into neural networks, enabling them to approximate complex mappings between inputs & outputs

- ReLU is preferred in hidden layers due to efficiency and better gradient flow.
- Softmax is essential for multi-class classification outputs.
- choice of activation function significantly impacts model performance.

DLT LAB-5.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[ ]

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten

# ------------------------------
# 1. Define activation functions
# ------------------------------
def sigmoid(x): return 1 / (1 + np.exp(-x))
def tanh(x): return np.tanh(x)
def relu(x): return np.maximum(0, x)
def leaky_relu(x): return np.where(x > 0, x, 0.01*x)
def elu(x, alpha=1.0): return np.where(x > 0, x, alpha*(np.exp(x)-1))

x = np.linspace(-6, 6, 400)

# Plot activation function curves
plt.figure(figsize=(12, 8))
plt.subplot(2,3,1); plt.plot(x, sigmoid(x)); plt.title("Sigmoid")
plt.subplot(2,3,2); plt.plot(x, tanh(x)); plt.title("Tanh")
plt.subplot(2,3,3); plt.plot(x, relu(x)); plt.title("ReLU")
plt.subplot(2,3,4); plt.plot(x, leaky_relu(x)); plt.title("Leaky ReLU")
plt.subplot(2,3,5); plt.plot(x, elu(x)); plt.title("ELU")
plt.tight_layout()
plt.show()

# ------------------------------
# 2. Load Dataset
# ------------------------------
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train/255.0, x_test/255.0

# ------------------------------
# 3. Build model function
# ------------------------------
def build_model(activation):
```

Variables Terminal

DLT LAB-5.ipynb

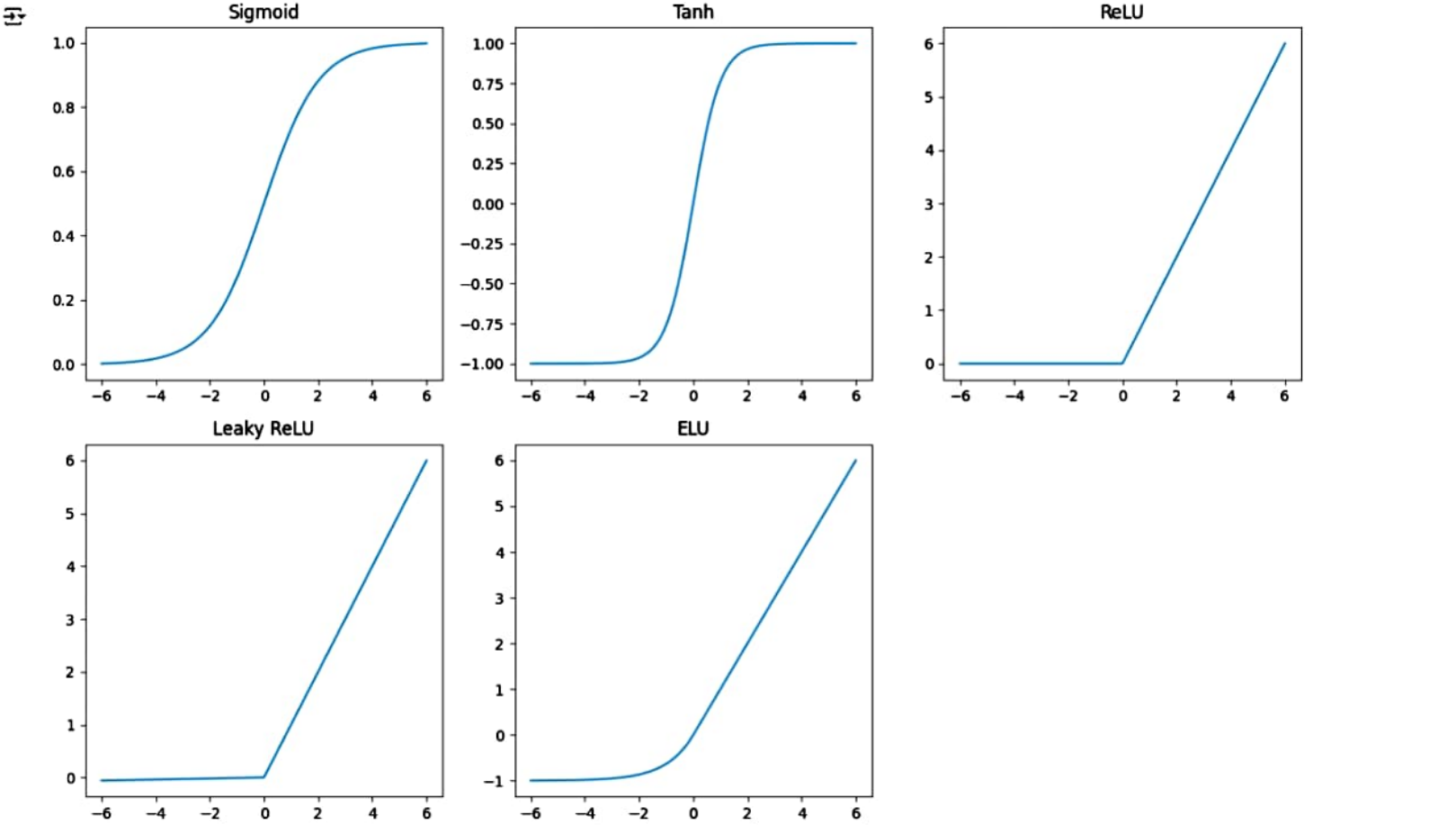
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

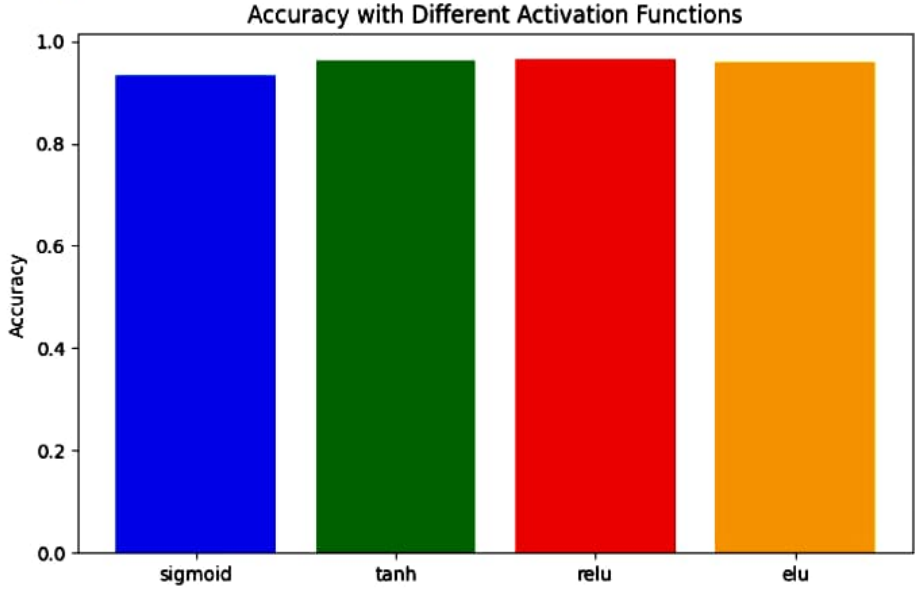
[ ]

```
def build_model(activation):
    model = Sequential([
        Flatten(input_shape=(28,28)),
        Dense(128, activation=activation),
        Dense(64, activation=activation),
        Dense(10, activation="softmax")
    ])
    model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
    return model

# ------------------------------
# 4. Train with different activations
# ------------------------------
activations = ['sigmoid', 'tanh', 'relu', 'elu']
results = {}

for act in activations:
    print(f"\nTraining with {act} activation...")
    model = build_model(act)
    model.fit(x_train, y_train, epochs=2, batch_size=128, verbose=0)
    loss, acc = model.evaluate(x_test, y_test, verbose=0)
    results[act] = acc

# ------------------------------
# 5. Plot comparison chart
# ------------------------------
plt.figure(figsize=(8,5))
plt.bar(results.keys(), results.values(), color=['blue','green','red','orange'])
plt.title("Accuracy with Different Activation Functions")
plt.ylabel("Accuracy")
plt.show()

# Print results
print("\nFinal Results:")
for act, acc in results.items():
    print(f"{act}: {acc*100:.2f}%")
```

Variables Terminal


Sigmoid
Tanh
ReLU
Leaky ReLU
ELU

```
Training with sigmoid activation...

Training with tanh activation...

Training with relu activation...

Training with elu activation...
```




```
Final Results:
sigmoid: 93.35%
tanh: 96.09%
relu: 96.58%
elu: 95.86%
```