Exp 4

Build a simple feed forward neural network to recognize hand written character

Aim: To build a simple feed forward neural network to recognize handwritten characters.

Objective:

1. To load and preprocess the MNIST dataset for neural network input
2. To build feed forward neural network model with hidden layers.
3. To train the model using Stochastic gradient descent optimizer and sparse categorical cross-entropy loss
4. Evaluate the trained model on test data and measure its accuracy
5. To predict the class of given handwritten image

Pseudo code:-

START

Load MNIST dataset (training and testing data)

Flatten each image from 28x28 to 784 features

Normalize pixel values to range (0, 1)

Create a sequential neural network:

Layer 1: Dense (128 neurons, ReLU activation)

Layer 2: Dense (64 neurons, ReLU activation)

Output layer: Dense(10 neurons, softmax activation)

Compile model:

Optimizer = stochastic gradient descent

Loss = sparse categorical crossentropy

Metric = accuracy

Train model on training data for 5 epochs

Evaluate model on testing data

Print test accuracy

END

Observation:-

→ The loss decreases with each epoch, showing that the model is learning

→ Accuracy improves steadily during training

Training:-

Epoch	Accuracy	Loss
1	0.9929	0.0232
2	0.9968	0.0128
3	0.9976	0.0099
4	0.9976	0.0088
5	0.9987	0.0058

Result:-

successfully implemented MNIST Dataset Trained on simple ANN ~~using~~ with 96.94% accuracy.

1/2/18/25

```python
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 for plotting only (optional)
# --------------------------------
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
# --------------------------------
```

```python
# --------------------------------
def build_model(activation):
    model = Sequential([
        Flatten(input_shape=(28,28)),
        Dense(128, activation=activation),
        Dense(64, activation=activation),
        Dense(10, activation="softmax")  # Softmax in output layer for multi-class classification
    ])
    model.compile(optimizer="adam",
                  loss="sparse_categorical_crossentropy",
                  metrics=["accuracy"])
    return model

# --------------------------------
# 4. Train with different activations (ELU removed)
# --------------------------------
activations = ['sigmoid', 'tanh', 'relu']
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'])
plt.title("Accuracy with Different Activation Functions")
plt.ylabel("Accuracy")
plt.show()

# Print results
```

Swayam | New tab | Reduce PPT file size | Convert JPG to PDF. Ima | New tab | Untitled0.ipynb - Colab

colab.research.google.com/drive/1F1iutjGVozUuXgPjePmVwpcNHzCJH-SB

School

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

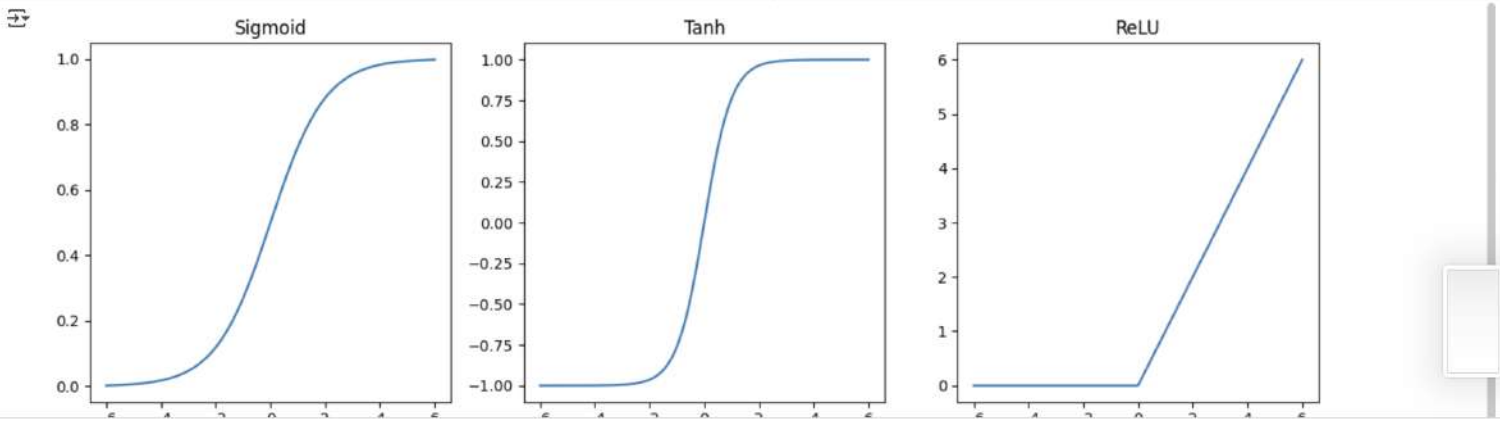
Share Gemini

Commands + Code + Text Run all Connect

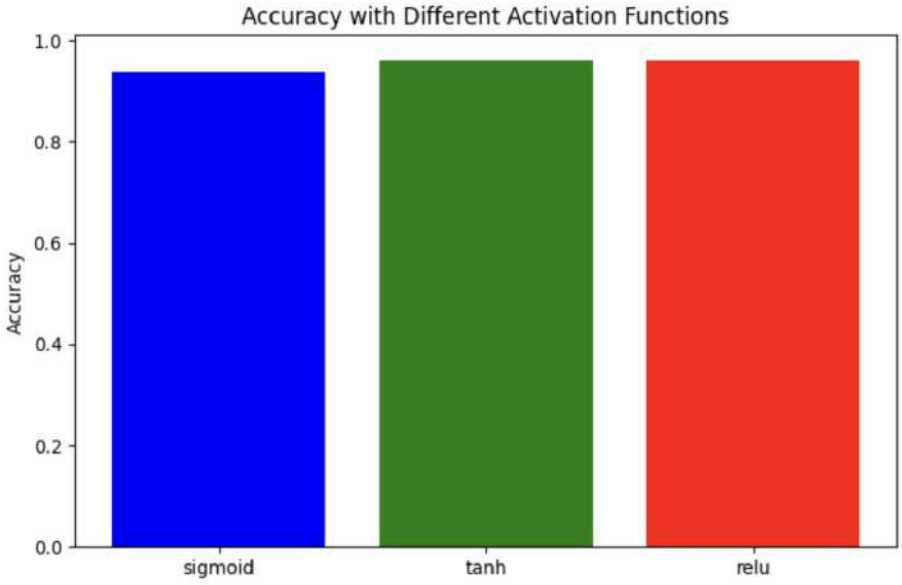
```
[ ]     results[act] = acc

    # ---------------------------------
    # 5. Plot comparison chart
    # ---------------------------------
    plt.figure(figsize=(8,5))
    plt.bar(results.keys(), results.values(), color=['blue', 'green', 'red'])
    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




```
Final Results:
sigmoid: 93.69%
tanh: 96.18%
relu: 96.00%
```