Exp-4
14-8-25

Build a simple Feed Forward Network To recognize handwritten character

Aim

To design, implement, and train a feed forward neural network using the MNIST dataset to recognize handwritten digit (0-9)

Objective

To understand the architecture of a simple FFNN

To preprocess image data appl. for training

To implement the network using dense (fully connected) layers with TensorFlow / keras.

To train the model and evaluate its performance on handwritten digit classification

To analysis the model's accuracy and generalization

Pseudocode

Begin

Import necessary libraries (TensorFlow, keras, Numpy)

Load the MNIST dataset (training and testing sets)

Normalize pixel value to range [0, 1]

Convert labels to one hot encoded vectors

Initialize a seq. neural network model

Add a Flatten layer to reshape 28x28 image into 1D vector

Add a Dense layer with 128 neurons and ReLU activation

Add a Dense layer with 64 neurons and ReLU activation

Add a Dense output layer with 10 neurons and softmax activation

Compile the model using Adam optimizer and categorical crossentropy loss

Train the model on the training data for 5 epochs. with validation split.

Observation :

ANN: on 8 epochs

Accuracy : 95.63%

Precision : 0.94

Recall : 0.93

f1 score : 0.93

ROC AUC : 0.985
(macro)

Evaluate the model on the test dataset
Display the test accuracy
Predict the classes of sample test image
END

Observation

Training accuracy consistntly improved across epochs
Validation accuracy was clos to training accuracy, suggesting minimal overfitting.
The model achieved high accuracy (above 95%) using only fully connected layer, without convolutional
Normalizing the input data sped up convergence during training.
The Adam optimizer was effective for this classification task.

Result: Successfully implemented MNIST Dataset trained on simple ANN.

1, PA/8/23

week3.py | week1.py | week2.py | jupyter-ra2311047010013@c

```
x_train = x_train / 255.0
x_test = x_test / 255.0

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Flatten(input_shape=(28, 28)),        # Flatten 28x28 image → 784 vector
    Dense(128, activation='relu'),        # Hidden layer with ReLU
    Dense(64, activation='relu'),         # Second hidden layer
    Dense(10, activation='softmax')       # Output layer (10 classes)
])

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

model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc*100:.2f}%")

import numpy as np

predictions = model.predict(x_test)
print("Predicted Label:", np.argmax(predictions[0]))

```

week3.py | week1.py | week2.py | jupyter-ra2311047010013@c

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

# Load dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize pixel values (0-255) → (0-1)
x_train = x_train / 255.0
x_test = x_test / 255.0

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Flatten(input_shape=(28, 28)),        # Flatten 28x28 image → 784 vector
    Dense(128, activation='relu'),        # Hidden layer with ReLU
    Dense(64, activation='relu'),         # Second hidden layer
    Dense(10, activation='softmax')       # Output layer (10 classes)
])

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