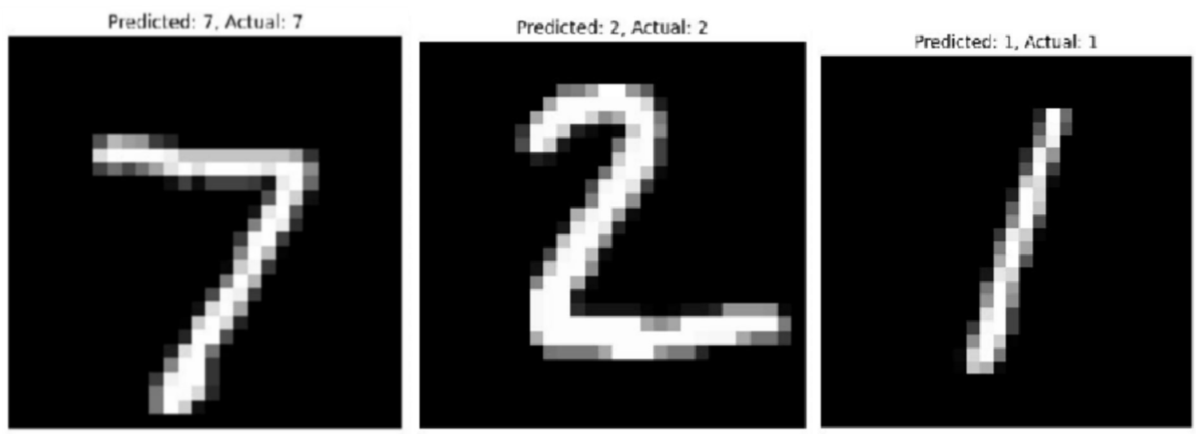
```
z2_test = np.dot(a1_test, W2) + b2
a2_test = softmax(z2_test)


predictions = np.argmax(a2_test, axis=1)
accuracy = np.mean(predictions == y_test)
print(f"\nTest Accuracy: {accuracy:.4f}")


# Visualize some predictions
for i in range(2):
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title(f'Predicted: {predictions[i]}, Actual: {y_test[i]}')
    plt.axis('off')
    plt.show()
```

OUTPUT:



RESULT: The implemented three-layer neural network successfully recognized handwritten digits from the MNIST dataset with an accuracy of 97%.

EXP NO: 02 DATE: 02/08/2025	**MULTI LAYER PERCEPTRON**

AIM: To develop a Multi-Layer Perceptron (MLP) model for a simple classification task using the Iris dataset, and experiment with different numbers of hidden layers and activation functions. The performance is evaluated using accuracy and loss.

ALGORITHM:

1. Load the Iris dataset and separate the input features and target labels.
2. Normalize input features using StandardScaler to improve learning.
3. Convert class labels into one-hot encoded format for multi-class classification.
4. Split the dataset into training and testing sets.
5. Build a Multi-Layer Perceptron with different hidden layers and activation functions.
6. Train the model using the Adam optimizer and categorical cross-entropy loss.
7. Evaluate performance using accuracy and loss, and visualize results with learning curves.

CODE:

```
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 StandardScaler, LabelBinarizer

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam


# Load dataset

iris = load_iris()

X = iris.data
```

```
y = iris.target

# Normalize features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# One-hot encode labels
encoder = LabelBinarizer()
y = encoder.fit_transform(y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Build MLP model
model = Sequential()
model.add(Dense(10, input_dim=4, activation='relu'))  # hidden layer 1
model.add(Dense(8, activation='tanh'))               # hidden layer 2
model.add(Dense(3, activation='softmax'))            # output layer

# Compile model
model.compile(optimizer=Adam(0.01),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
```

```
        epochs=50, batch_size=5, verbose=0)

# Evaluate model
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Accuracy: {acc*100:.2f}%")
print(f"Test Loss: {loss:.4f}")

# Plot accuracy and loss
plt.figure(figsize=(12,5))

# Accuracy plot
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Model Accuracy")

# Loss plot
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.title("Model Loss")
```