Implement a Basic Artificial Neural Network Problem Statement: Build a simple feed-forward neural network to classify handwritten digits from the MNIST dataset. Activities: • Use TensorFlow/Keras • Train and evaluate ANN • Display sample predictions

```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np

# 1️⃣ Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = datasets.mnist.load_data()

# 2️⃣ Normalize the pixel values (0-255 -> 0-1)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Flatten 28x28 images into 784-dimensional vectors
x_train = x_train.reshape(-1, 28*28)
x_test = x_test.reshape(-1, 28*28)

print("Training data shape:", x_train.shape)
print("Test data shape:", x_test.shape)

# 3️⃣ Build the ANN model
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(784,)),  # Hidden layer 1
    layers.Dense(64, activation='relu'),                       # Hidden layer 2
    layers.Dense(10, activation='softmax')                     # Output layer (10 digits)
])

# 4️⃣ Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 5️⃣ Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.1)

# 6️⃣ Evaluate the model on test data
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"\nTest Accuracy: {test_acc:.4f}")

# 7️⃣ Display sample predictions
num_samples = 10
sample_images = x_test[:num_samples].reshape(-1, 28, 28)
sample_labels = y_test[:num_samples]
predictions = np.argmax(model.predict(x_test[:num_samples]), axis=1)

plt.figure(figsize=(10, 3))
for i in range(num_samples):
    plt.subplot(2, 5, i + 1)
    plt.imshow(sample_images[i], cmap='gray')
    plt.title(f"True: {sample_labels[i]}\nPred: {predictions[i]}")
    plt.axis('off')
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Training data shape: (60000, 784)
Test data shape: (10000, 784)
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_sha
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.8697 - loss: 0.4478 - val_accuracy: 0.9707 - val_loss:
Epoch 2/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.9670 - loss: 0.1119 - val_accuracy: 0.9748 - val_loss: 0
Epoch 3/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.9794 - loss: 0.0696 - val_accuracy: 0.9770 - val_loss: 0
Epoch 4/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9832 - loss: 0.0526 - val_accuracy: 0.9778 - val_loss: 0
Epoch 5/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.9866 - loss: 0.0398 - val_accuracy: 0.9772 - val_loss: 0
Epoch 6/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9896 - loss: 0.0312 - val_accuracy: 0.9787 - val_loss: 0
Epoch 7/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.9914 - loss: 0.0253 - val_accuracy: 0.9798 - val_loss: 0
Epoch 8/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.9925 - loss: 0.0226 - val_accuracy: 0.9798 - val_loss: 0
Epoch 9/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9927 - loss: 0.0208 - val_accuracy: 0.9797 - val_loss: 0
Epoch 10/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.9955 - loss: 0.0141 - val_accuracy: 0.9745 - val_loss:
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9710 - loss: 0.1354

Test Accuracy: 0.9746
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
```