Ass3

```
# a. LOADING AND PREPROCESSING THE IMAGE DATA

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import random
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Conv2D, Dense, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.datasets import mnist

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)

# Normalize pixel values to range [0, 1]
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# Reshape data to include channel dimension
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1))
X_test = X_test.reshape((X_test.shape[0], 28, 28, 1))

# Plot sample images
def plot_digit(image, digit, plt, i):
    plt.subplot(4, 5, i + 1)
    plt.imshow(image.squeeze(), cmap='gray')
    plt.title(f"Digit: {digit}")
    plt.axis('off')

plt.figure(figsize=(12, 8))
for i in range(20):
    plot_digit(X_train[i], y_train[i], plt, i)
plt.show()

# Split training data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42
)

# ============================================================
# b. DEFINING THE MODEL'S ARCHITECTURE
# ============================================================
model = Sequential([
    Conv2D(32, (3, 3), activation="relu", input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(100, activation="relu"),
    Dense(10, activation="softmax")
])

# Compile model
optimizer = SGD(learning_rate=0.01, momentum=0.9)
model.compile(optimizer=optimizer,
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
model.summary()

# ============================================================
# c. TRAINING THE MODEL
# ============================================================
Model_log = model.fit(
    X_train,
    y_train,
    epochs=10,
    batch_size=15,
    verbose=1,
    validation_data=(X_val, y_val)
)
```

```
# ============================================================
# d. ESTIMATING THE MODEL'S PERFORMANCE
# ============================================================
# Evaluate on test set
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f"\nTest Accuracy: {test_acc * 100:.2f}%")

# Predict and visualize results on random test samples
plt.figure(figsize=(12, 8))
for i in range(20):
    image = random.choice(X_test)
    true_label = y_test[random.randint(0, len(y_test) - 1)]
    pred_label = np.argmax(model.predict(image.reshape((1, 28, 28, 1))), axis=-1)[0]
    plot_digit(image, f"Pred: {pred_label}", plt, i)
plt.show()

# Compute accuracy using sklearn
predictions = np.argmax(model.predict(X_test), axis=-1)
print("Accuracy Score (sklearn):", accuracy_score(y_test, predictions))

# Display a random test image
n = random.randint(0, len(X_test) - 1)
plt.imshow(X_test[n].squeeze(), cmap='gray')
plt.title(f"Actual: {y_test[n]}, Predicted: {predictions[n]}")
plt.axis('off')
plt.show()
```