Aim: MNIST Handwritten character detection using PyTorch, Keras and Tensorflow.

Software requirement for Python: Jupyter Notebook

Theory:

1. The steps for MNIST Handwritten character detection using PyTorch, Keras, and Tensorflow are similar and involve the following:
2. Load the dataset: The first step is to load the MNIST dataset, which contains 60,000 training images and 10,000 test images of handwritten digits.
3. Prepare the data: In this step, you will normalize the data and convert the labels to one-hot vectors.
4. Build the model: You will build a neural network model with input layer, hidden layers, and output layer.
5. Compile the model: You will compile the model by specifying the loss function, optimizer, and metrics.
6. Train the model: In this step, you will train the model on the training dataset using the fit method.
7. Evaluate the model: After training, you will evaluate the performance of the model on the test dataset using the evaluate method.
8. Make predictions: Finally, you will make predictions on new data using the predict method.

Procedure:

1. This code implements a neural network model using the Keras API in TensorFlow for the MNIST handwritten digit recognition task. The code first loads and preprocesses the MNIST dataset by scaling the pixel values to the range [0, 1].
2. Then, a simple feedforward neural network with one hidden layer is defined using the Sequential model. The input layer is a Flatten layer that converts the 28x28 input images to a 784-dimensional vector. The hidden layer has 128 units with the ReLU activation function. The output layer has 10 units with the softmax activation function that outputs the probability distribution over 10 classes.
3. The model is compiled using the Adam optimizer with a learning rate of 0.001 and the sparse categorical cross-entropy loss function. The accuracy metric is also specified for evaluation.
4. The model is then trained on the training set using the fit() method with a batch size of 128 and for 10 epochs.
5. Finally, the model is evaluated on the test set using the evaluate() method, and the test loss and accuracy are printed.

Conclusion: Thus, we have studied MNIST Handwritten character detection using PyTorch, Keras and Tensorflow.

Program Code and Output

```
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.optimizers import Adam
# Load and preprocess the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0

# Define the neural network architecture
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
# Train the model
model.fit(x_train, y_train, batch_size=128, epochs=10, verbose=1)

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_acc:.4f}')
```

OUTPUT OF THE CODE

```
Epoch 1/10

469/469 [==============================] - 2s 3ms/step - loss: 0.3584 - accuracy: 0.9014

Epoch 2/10

469/469 [==============================] - 1s 3ms/step - loss: 0.1667 - accuracy: 0.9522

Epoch 3/10

469/469 [==============================] - 1s 3ms/step - loss: 0.1191 - accuracy: 0.9651

Epoch 4/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0934 - accuracy: 0.9728

Epoch 5/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0757 - accuracy: 0.9784

Epoch 6/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0632 - accuracy: 0.9817

Epoch 7/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0526 - accuracy: 0.9850

Epoch 8/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0453 - accuracy: 0.9872

Epoch 9/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0378 - accuracy: 0.9896

Epoch 10/10

469/469 [==============================] - 1s 3ms/step - loss: 0.0323 - accuracy: 0.9913

313/313 - 1s - loss: 0.0716 - accuracy: 0.9782 - 613ms/epoch - 2ms/step

Test Loss: 0.0716

Test Accuracy: 0.9782
```