PROGRAM

```
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
import numpy as np
import warnings
warnings.filterwarnings('ignore')

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train.shape, X_test.shape
IMG_SIZE = 32

import cv2
def resize(img_array):
    tmp = np.empty((img_array.shape[0], IMG_SIZE, IMG_SIZE))

    for i in range(len(img_array)):
        img = img_array[i].reshape(28, 28).astype('uint8')
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        img = img.astype('float32')/255
        tmp[i] = img
    return tmp

X_train_resize = resize(X_train)
X_test_resize = resize(X_test)

x_train_final = np.stack((X_train_resize,)*3, axis=-1)
x_test_final = np.stack((X_test_resize,)*3, axis=-1)
print(x_train_final.shape)
print(x_test_final.shape)

from keras.utils import to_categorical
y_train_final = to_categorical(y_train, num_classes=10)
print(y_train_final.shape)
y_test_final = to_categorical(y_test, num_classes=10)
print(y_test_final.shape)
y_train_final = to_categorical(y_train)
y_test_final = to_categorical(y_test)

print(y_train_final.shape)
print(y_test_final.shape)

from keras.models import Sequential
from keras.applications import VGG19
from keras.layers import Dense, Flatten
```

```
vgg19 = VGG19(weights = 'imagenet',
              include_top = False,
              input_shape=(IMG_SIZE, IMG_SIZE, 3)
              )

model = Sequential()
model.add(vgg19)
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

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

history = model.fit(x_train_final, y_train_final,
                    epochs=5,
                    batch_size=128,
                    validation_data=(x_test_final, y_test_final))

test_loss, test_accuracy = model.evaluate(x_test_final, y_test_final)
print("Loss = %.2f"%test_loss)
print("Accuracy=%.2f"%test_accuracy)

preds = model.predict(x_test_final, batch_size=128)
preds.shape
results = np.argmax(preds, axis=-1)
results.shape
history.history.keys()

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss and Accuracy')
plt.xlabel('no.of epochs')
plt.ylabel('Accuracy/Loss')
plt.legend(['accuracy', 'val_accuracy', 'loss', 'val_loss'])
plt.show()
```

OUTPUT

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
(60000, 32, 32, 3)
(10000, 32, 32, 3)
(60000, 10)
(10000, 10)
(60000, 10)
(10000, 10)
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_not
80134624/80134624 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 1, 1, 512)         20024384

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 10)                5130

=================================================================
Total params: 20029514 (76.41 MB)
Trainable params: 20029514 (76.41 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/5
469/469 [==============================] - 52s 69ms/step - loss: 0.8508 - accuracy: 0.7014 - val_loss: 0.1816 - val_accuracy: 0.9595
Epoch 2/5
469/469 [==============================] - 32s 69ms/step - loss: 0.1034 - accuracy: 0.9749 - val_loss: 0.0603 - val_accuracy: 0.9851
Epoch 3/5
469/469 [==============================] - 32s 67ms/step - loss: 0.0706 - accuracy: 0.9832 - val_loss: 0.0867 - val_accuracy: 0.9792
Epoch 4/5
469/469 [==============================] - 32s 68ms/step - loss: 0.0553 - accuracy: 0.9869 - val_loss: 0.0655 - val_accuracy: 0.9853
Epoch 5/5
469/469 [==============================] - 32s 69ms/step - loss: 0.0604 - accuracy: 0.9869 - val_loss: 0.0508 - val_accuracy: 0.9881
313/313 [==============================] - 3s 10ms/step - loss: 0.0508 - accuracy: 0.9881

Loss = 0.05
Accuracy=0.99
79/79 [==============================] - 1s 17ms/step
```