In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-py
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all fi

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets prese
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside
```

In [ ]:
```
import tensorflow as tf
from matplotlib import pyplot as plt
import numpy as np
```

In [ ]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
assert x_train.shape == (60000, 28, 28)
assert x_test.shape == (10000, 28, 28)
assert y_train.shape == (60000,)
y_train = tf.keras.utils.to_categorical(y_train,10)
assert y_test.shape == (10000,)
y_test = tf.keras.utils.to_categorical(y_test,10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
st.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

In [ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten,Dense
model=Sequential()
input_layer=Flatten(input_shape=(28,28))
model.add(input_layer)
layer1=Dense(128,activation="relu")
model.add(layer1)
layer2=Dense(128,activation="relu")
model.add(layer2)
output_layer=Dense(10,activation="softmax")
model.add(output_layer)
```

```
2022-09-19 15:00:58.117890: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:00:58.260727: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:00:58.264807: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:00:58.269508: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-09-19 15:00:58.269968: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:00:58.274297: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:00:58.278325: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:01:01.389824: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:01:01.394030: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:01:01.398139: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-09-19 15:01:01.402395: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 15401 MB memory:  -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
```

In [ ]:
```
model.compile(optimizer = 'adam',
loss = 'categorical_crossentropy',
metrics=['accuracy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model.fit(x_train,y_train,epochs=10)
```

```
2022-09-19 15:01:02.475504: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/10
1875/1875 [==============================] - 7s 3ms/step - loss: 1.5202 - accuracy: 0.8782
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3028 - accuracy: 0.9314
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2130 - accuracy: 0.9440
Epoch 4/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1641 - accuracy: 0.9554
Epoch 5/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1407 - accuracy: 0.9595
Epoch 6/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1232 - accuracy: 0.9650
Epoch 7/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1129 - accuracy: 0.9682
Epoch 8/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1068 - accuracy: 0.9700
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0955 - accuracy: 0.9740
Epoch 10/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0910 - accuracy: 0.9753
```

Out[ ]:
```
<keras.callbacks.History at 0x7f5b7d002f90>
```

In [ ]:
```
model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 2s 4ms/step - loss: 0.1851 - accuracy: 0.9594
```

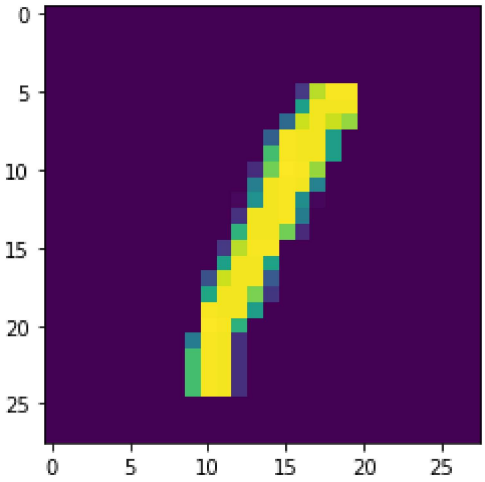
Out[ ]: [0.1850961446762085, 0.9593999981880188]

In [ ]:
```
x_test[77].shape
```

Out[ ]: (28, 28)

In [ ]:
```
import matplotlib.pyplot as plt
img = np.reshape(x_train[77], (28,28))
plt.imshow(img)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f5b73a4e750>



In [ ]:
```
model.predict(x_test[77].reshape(-1,784))
```

Out[ ]:
```
array([[2.8925051e-08, 4.5342877e-04, 9.7698683e-01, 4.0163308e-05,
        3.2951499e-07, 1.5843257e-10, 4.6248314e-07, 2.2500431e-02,
        1.4835601e-05, 3.6079748e-06]], dtype=float32)
```

In [ ]:
```
y_test[77]
```

Out[ ]:
```
array([0., 0., 1., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

In [ ]:
```
y_train[77]
```

Out[ ]:
```
array([0., 1., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

In [ ]: