```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
import tensorflow as tf
import cv2
import numpy as np

(X_train, y_train), (X_test, y_test)=tf.keras.datasets.mnist.load_data()
X_train = X_train/255.0
X_test = X_test/255.0

digit_classifier=Sequential()
digit_classifier.add(Conv2D(64,(3,3),input_shape=(28,28,1),activation='relu'))
#by default the stride is 1
digit_classifier.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
digit_classifier.add(Dropout(0.2))
digit_classifier.add(Conv2D(64,(3,3),activation='relu'))
digit_classifier.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
digit_classifier.add(Dropout(0.2))
digit_classifier.add(Flatten())

digit_classifier.add(Dense(units=256,activation='relu'))
digit_classifier.add(Dropout(0.5))
digit_classifier.add(Dense(units=10,activation='softmax'))
digit_classifier.compile(optimizer='adam',loss=tf.keras.losses.sparse_categorical_
digit_classifier.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_7 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               409856
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                2570
=================================================================
Total params: 449,994
Trainable params: 449,994
Non-trainable params: 0
_________________________________________________________________
```

Always use GPU to build CNN model

```
%%time
digit_classifier.fit(X_train.reshape(-1,28,28,1),y_train,epochs=10,batch_size=60)
```

```
Epoch 1/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.2092 - acc
Epoch 2/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0716 - acc
Epoch 3/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0574 - acc
Epoch 4/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0451 - acc
Epoch 5/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0407 - acc
Epoch 6/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0382 - acc
Epoch 7/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0315 - acc
Epoch 8/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0292 - acc
Epoch 9/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0286 - acc
Epoch 10/10
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0247 - acc
CPU times: user 59.1 s, sys: 16.3 s, total: 1min 15s
Wall time: 1min 11s
<tensorflow.python.keras.callbacks.History at 0x7fc30ccea390>
```

```
#Testing the classifire
score =digit_classifier.evaluate(X_test.reshape(-1,28,28,1),y_test)
print(f"Test score : {score[1]}")
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0236 - accur
Test score : 0.9933000206947327
```

```
gray=cv2.imread('0.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([0])
```

```
gray=cv2.imread('1.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([1])
```

```
gray=cv2.imread('2.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([2])
```

```
gray=cv2.imread('3.jpg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([3])
```

```
gray=cv2.imread('4.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([4])
```

```
gray=cv2.imread('5.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([5])
```

```
gray=cv2.imread('6.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([6])
```

```
gray=cv2.imread('8.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([8])
```

```
gray=cv2.imread('9.jpeg',0)
gray=gray.reshape(1,-1)
gray=gray/255.0
np.argmax(digit_classifier.predict(gray.reshape(-1,28,28,1)), axis=-1)
```

```
array([2])
```