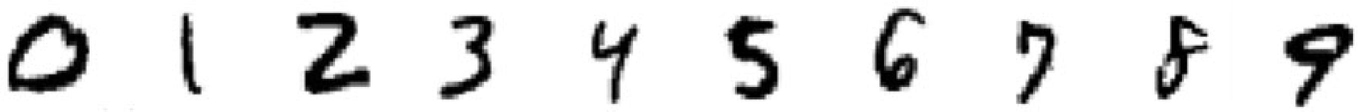
Predicting **Handwritten digits**

Deep Learning project by Ayushman Rayaguru

Handwritten Digit Recognition is an interesting machine learning problem in which we have to identify the handwritten digits through various classification algorithms. There are a number of ways and algorithms to recognize handwritten digits, including Deep Learning/**CNN**, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc.



```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
import numpy as np

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)
```

```
    (60000, 28, 28) (60000,)
```

Preprocessing data

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
num_classes = 10
import tensorflow as tf
# convert class vectors to binary class matrices
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
```

```
    10000 test samples
```

Let's design our **CNN model**

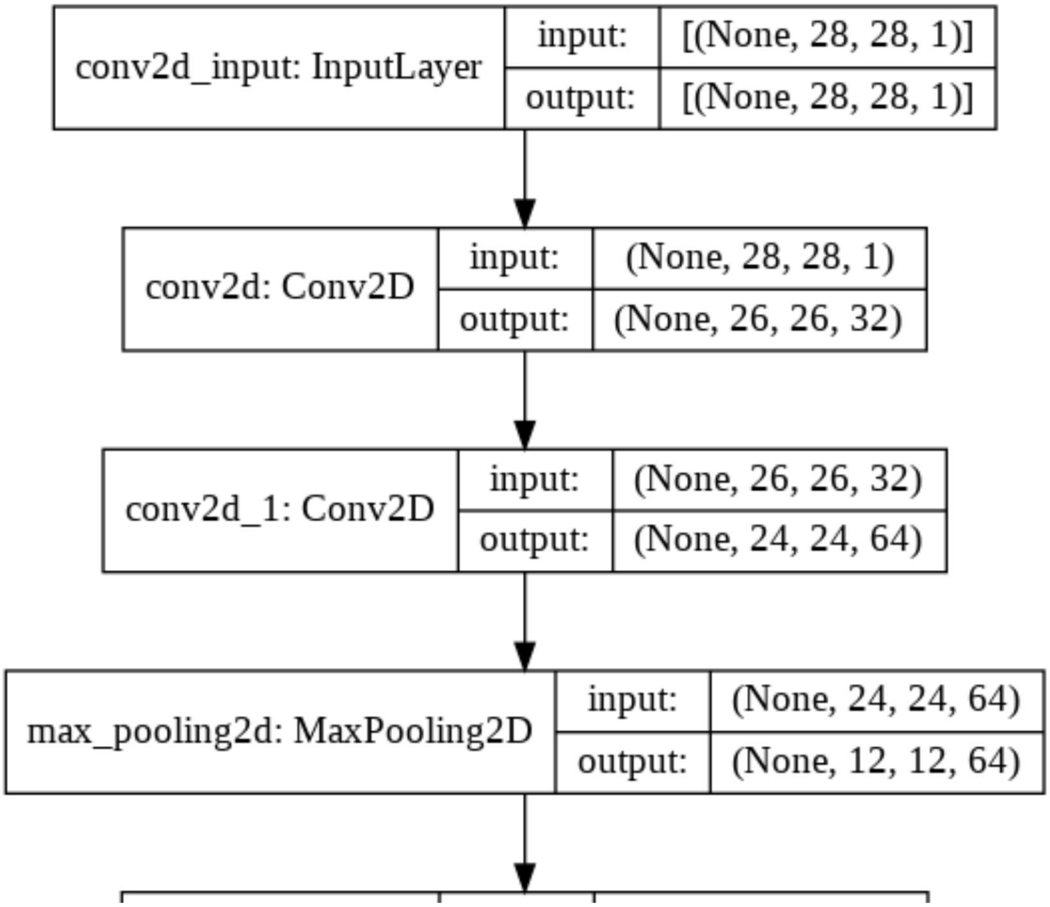
```
batch_size = 128
num_classes = 10
epochs = 10

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers
```

Model in a glance

```
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True
```

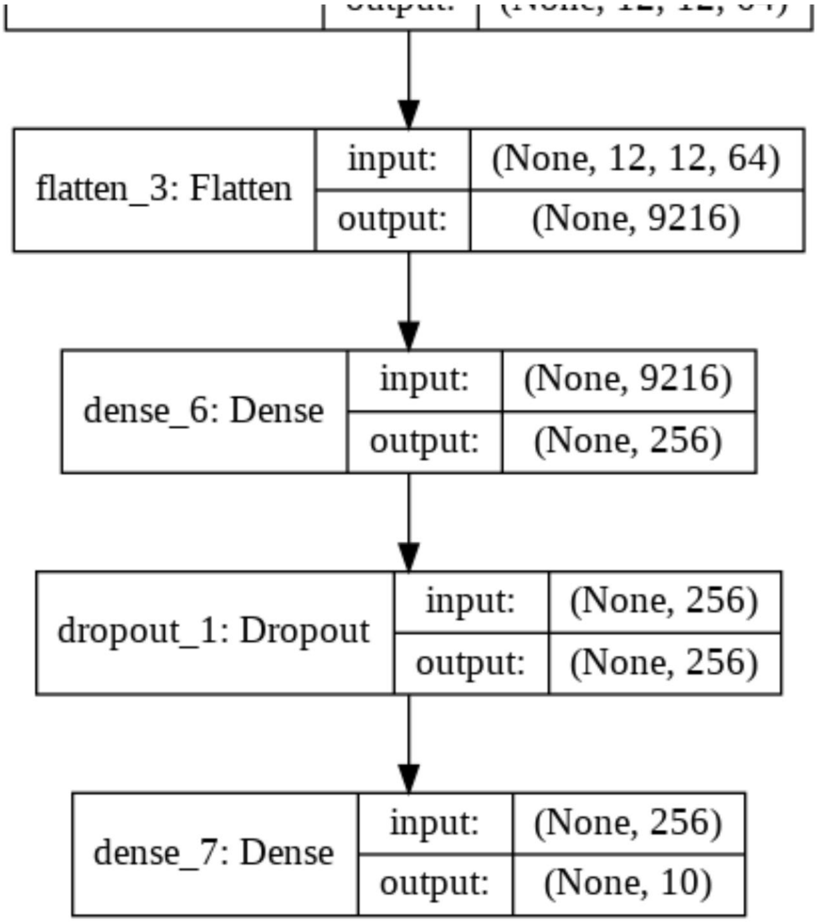


conv2d_input: InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]

conv2d: Conv2D | input: (None, 28, 28, 1) | output: (None, 26, 26, 32)

conv2d_1: Conv2D | input: (None, 26, 26, 32) | output: (None, 24, 24, 64)

max_pooling2d: MaxPooling2D | input: (None, 24, 24, 64) | output: (None, 12, 12, 64)



Training the model

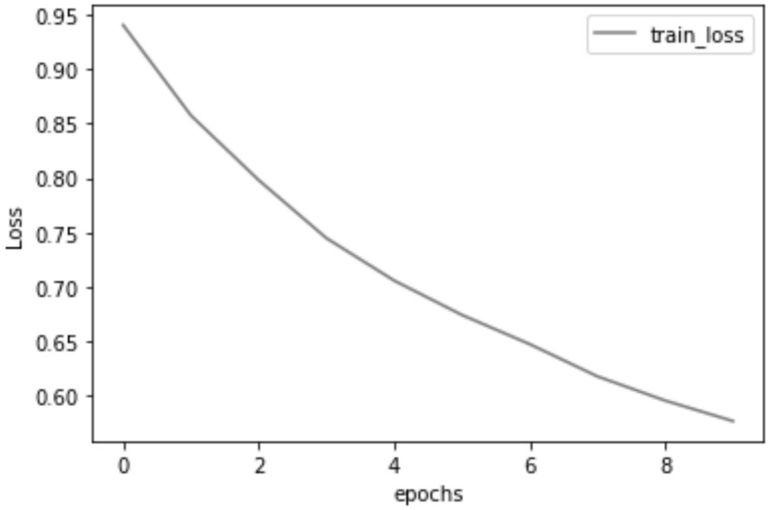
```
from keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0
history = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,
print("The model has successfully trained")

model.save('digits_recog.h5')
print("Saving the model as digits_recog.h5")
```

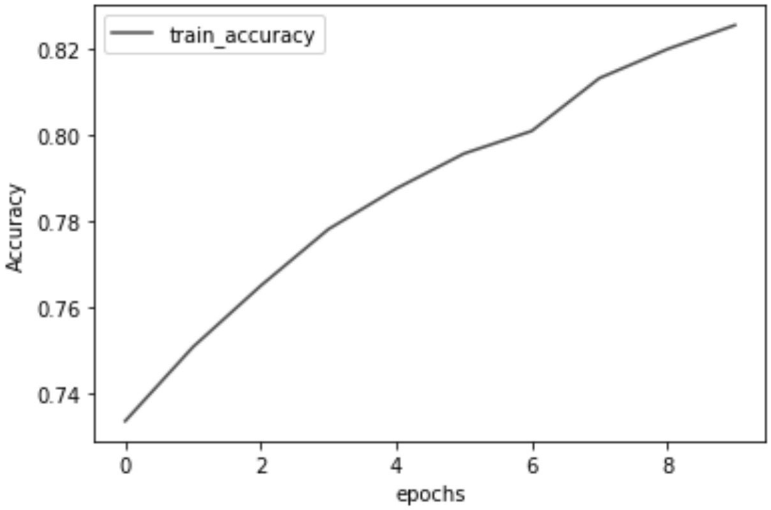
```
    Epoch 1/10
    469/469 [==============================] - 8s 17ms/step - loss: 0.9403 - accur
    Epoch 2/10
    469/469 [==============================] - 8s 17ms/step - loss: 0.8575 - accur
    Epoch 3/10
    469/469 [==============================] - 8s 17ms/step - loss: 0.7982 - accur
    Epoch 4/10
    469/469 [==============================] - 8s 16ms/step - loss: 0.7449 - accur
    Epoch 5/10
    469/469 [==============================] - 8s 16ms/step - loss: 0.7060 - accur
    Epoch 6/10
    469/469 [==============================] - 8s 16ms/step - loss: 0.6744 - accur
    Epoch 7/10
    469/469 [==============================] - 8s 16ms/step - loss: 0.6477 - accur
    Saving the model as digits_recog.h5
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.xlabel('epochs')
plt.ylabel('Loss')
plt.legend(['train_loss','val_loss'], loc=0)
```

```
    <matplotlib.legend.Legend at 0x7f59a0258110>
```



```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],'b-')
plt.xlabel('epochs')
plt.ylabel('Accuracy')
plt.legend(['train_accuracy','val_accuracy'], loc=0)
plt.show()
```



Evaluating the model

*Great our model has **accuracy of 90%**.It's ready for its deployment using tkinter*

Can accuracy be better ?

Lets try making model without use of convolution neural network, rather we will proceed with relu activation function and softmax as last layer.

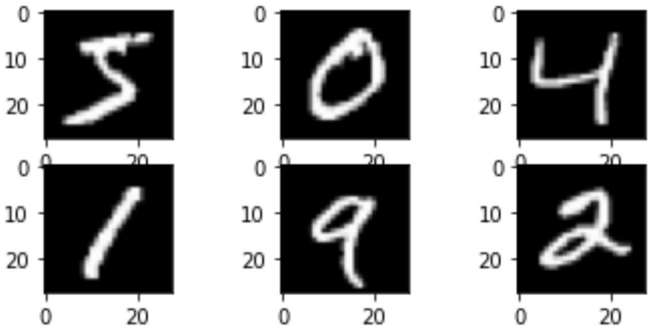
Load Data

```
import tensorflow as tf
obj = tf.keras.datasets.mnist
(train_x,train_y), (test_x, test_y) = obj.load_data()
```

Using same data

Having a look at the images

```
import matplotlib.pyplot as plt
for i in range(6):
  plt.subplot(330 + 1 +i)
  plt.imshow(train_x[i],'gray')
```



Normalizing the training and testing set

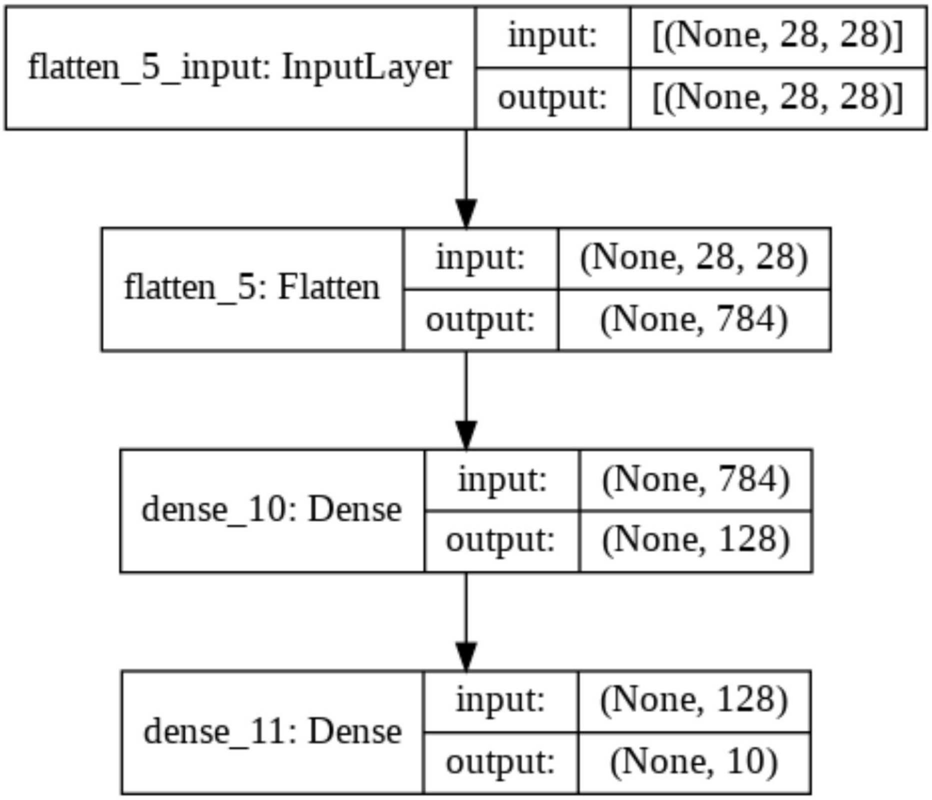
```
train_x = train_x/255
```

```
model2.add(Dense(128, activation='relu'))
model2.add(Dense(10, activation ='softmax'))
```

```
model2.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['ac
```

Quick glance on model 2

```
from keras.utils.vis_utils import plot_model
plot_model(model2, to_file='model2_plot.png', show_shapes=True, show_layer_names=Tr
```



flatten_5_input: InputLayer | input: [(None, 28, 28)] | output: [(None, 28, 28)]

flatten_5: Flatten | input: (None, 28, 28) | output: (None, 784)

dense_10: Dense | input: (None, 784) | output: (None, 128)

dense_11: Dense | input: (None, 128) | output: (None, 10)

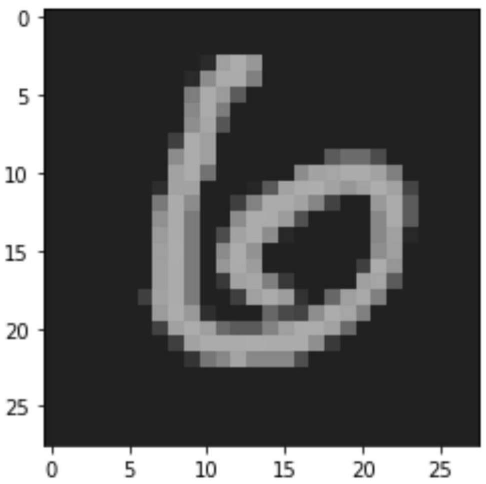
Fit the model to training data

```
model2.fit(train_x, train_y, epochs = 5)
model2.save('digits_4layer.h5')
print("Saving the model as digits_4layer.h5")
print(model2.evaluate(test_x,test_y))
```

```
313/313 [==============================] - 1s 2ms/step - loss: 3968.3496 - acc
[3968.349609375, 0.09669999778270721]
```

```
plt.imshow(test_x[11])
prediction = model2.predict(test_x)
print(np.argmax(prediction[11]))
```

```
6
```



Great our model did great work it works with **accuracy of 96.69**