In [ ]:

```
# Importing Libraries

import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.datasets import mnist

from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
```

In [ ]:

```
mnist.load_data?
```

In [ ]:

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/
mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

In [ ]:

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[ ]:

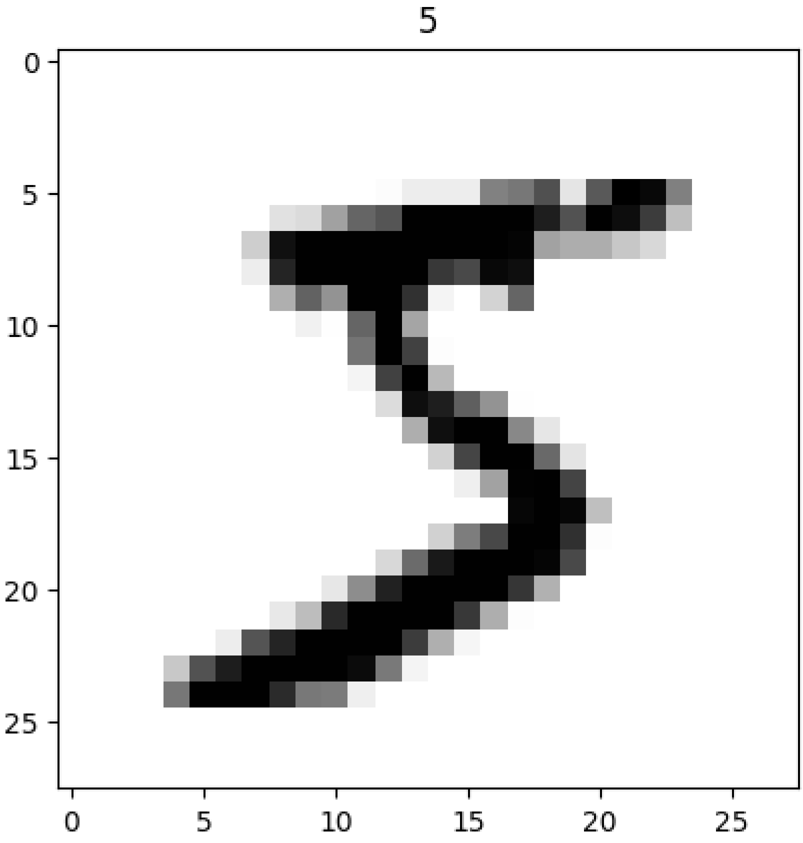
```
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

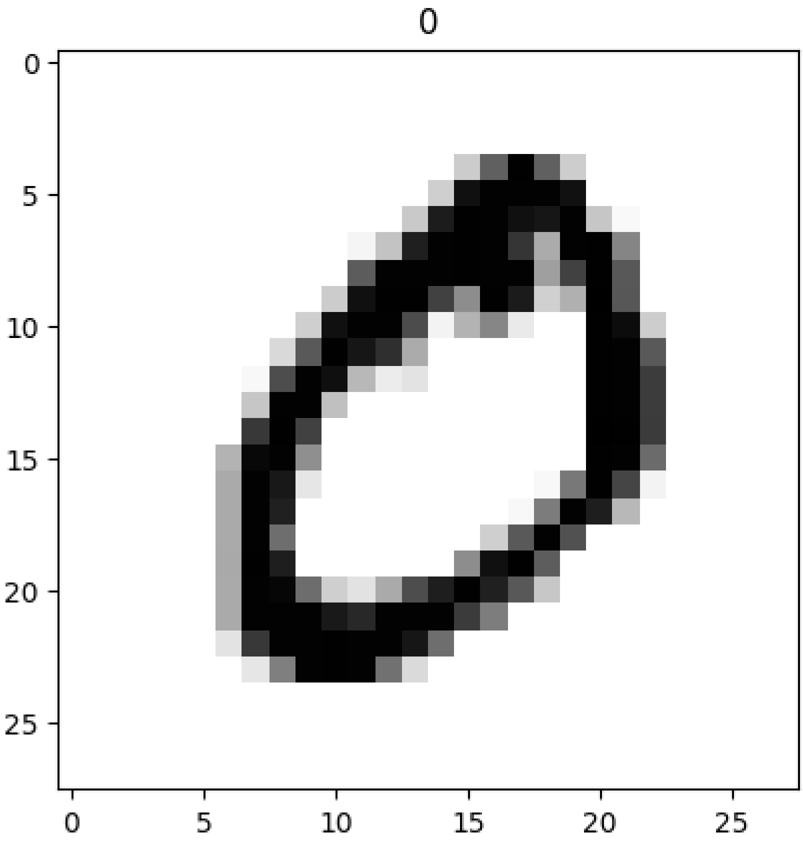
In [ ]:

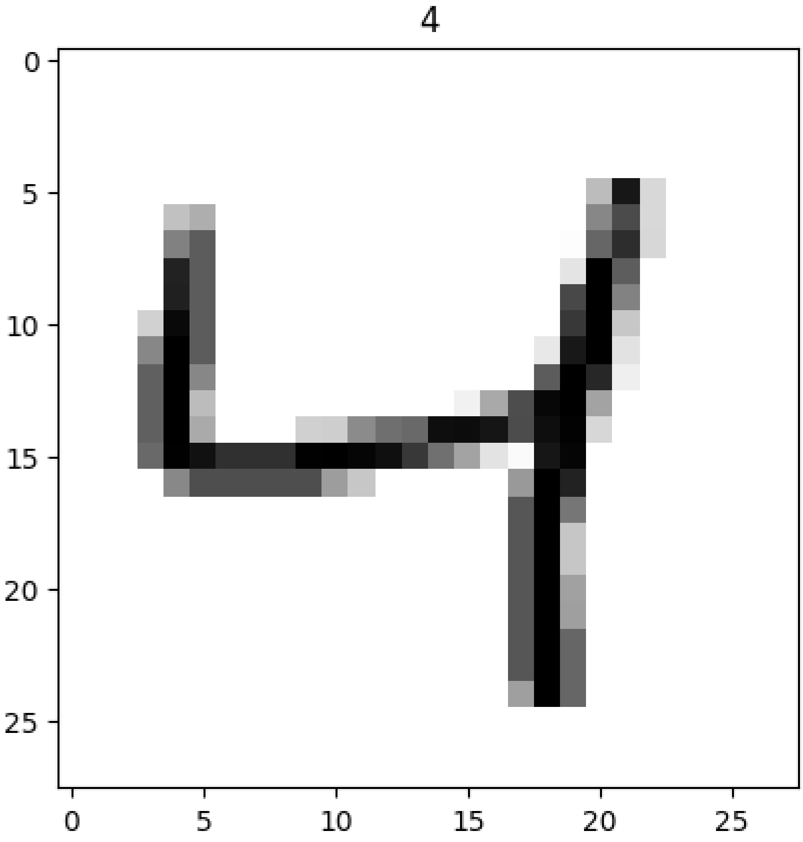
```
def plot_input_img(i):
  plt.imshow(X_train[i], cmap = 'binary')
  plt.title(y_train[i])
  plt.show()
```

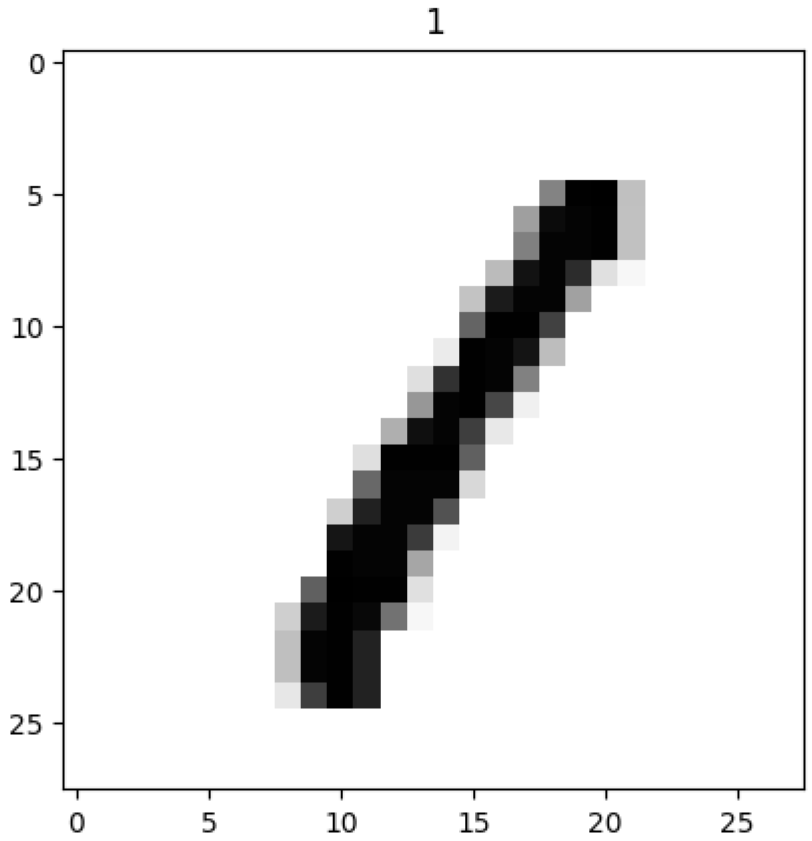
In [ ]:

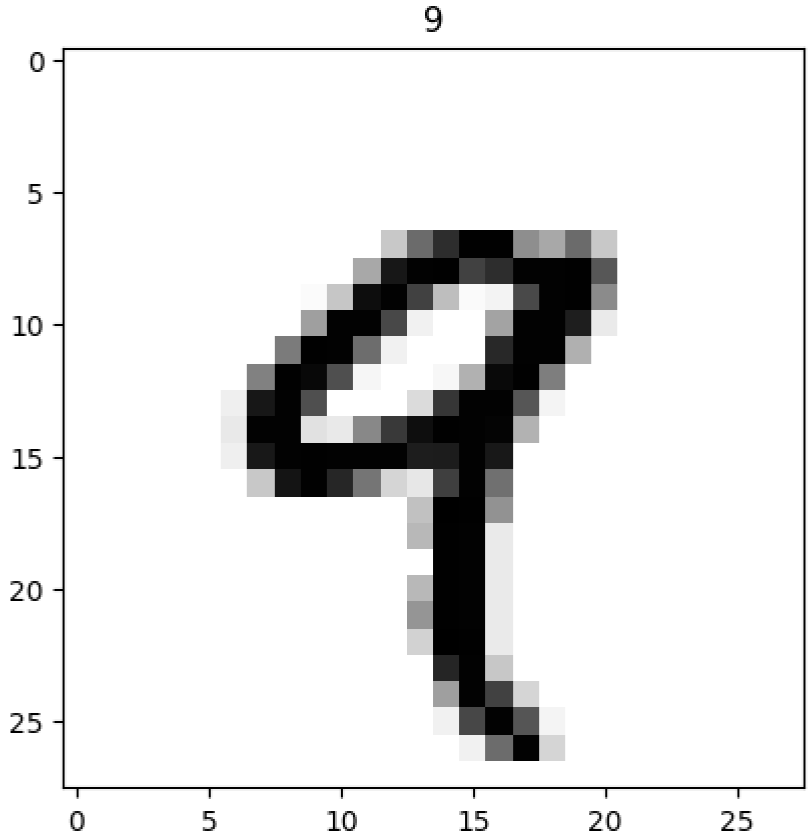
```
for i in range(10):
  plot_input_img(i)
```

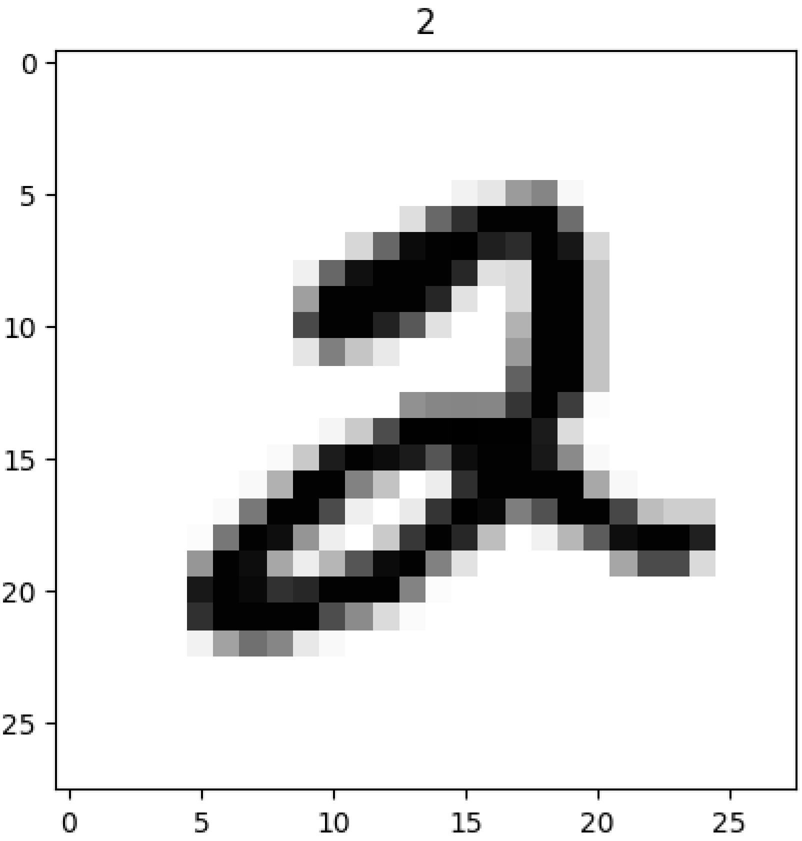


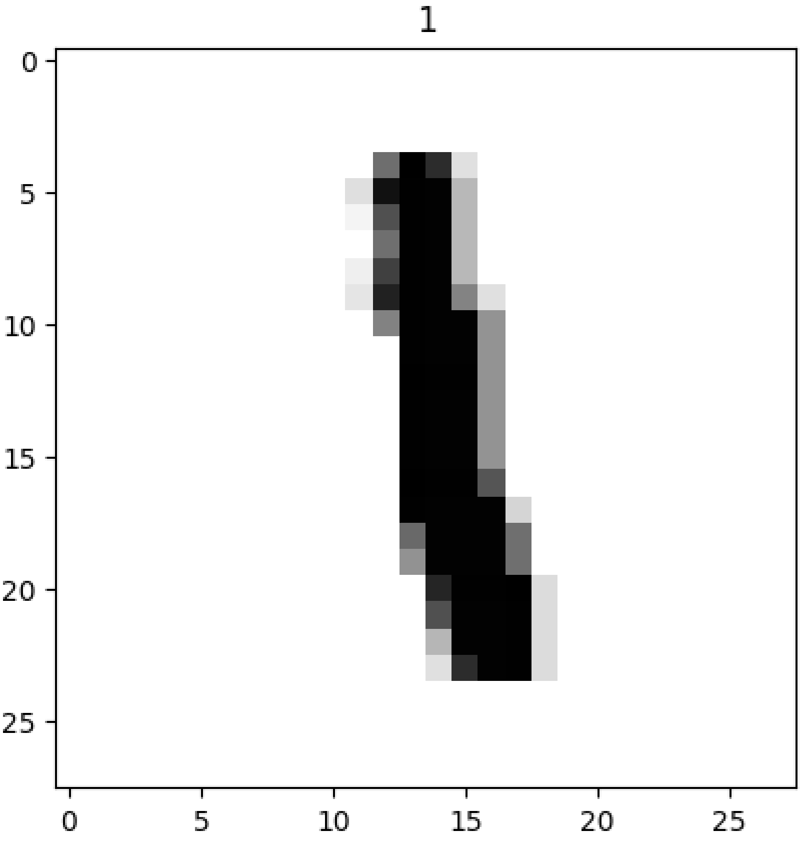

0
0
5
10
15
20
25
0
5
10
15
20
25

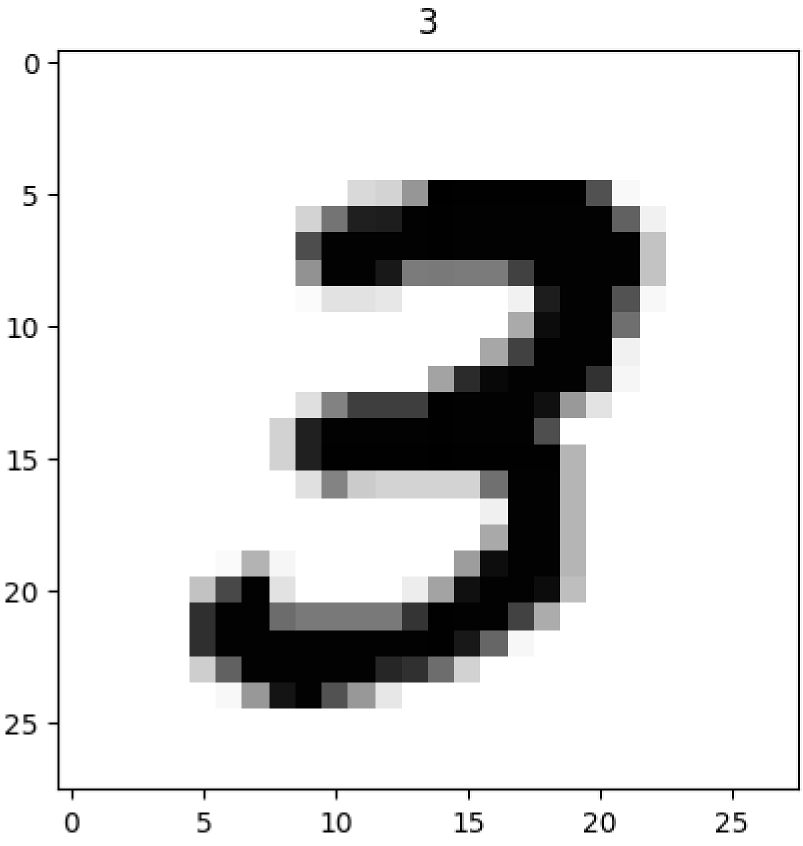

4
0
5
10
15
20
25
0
5
10
15
20
25

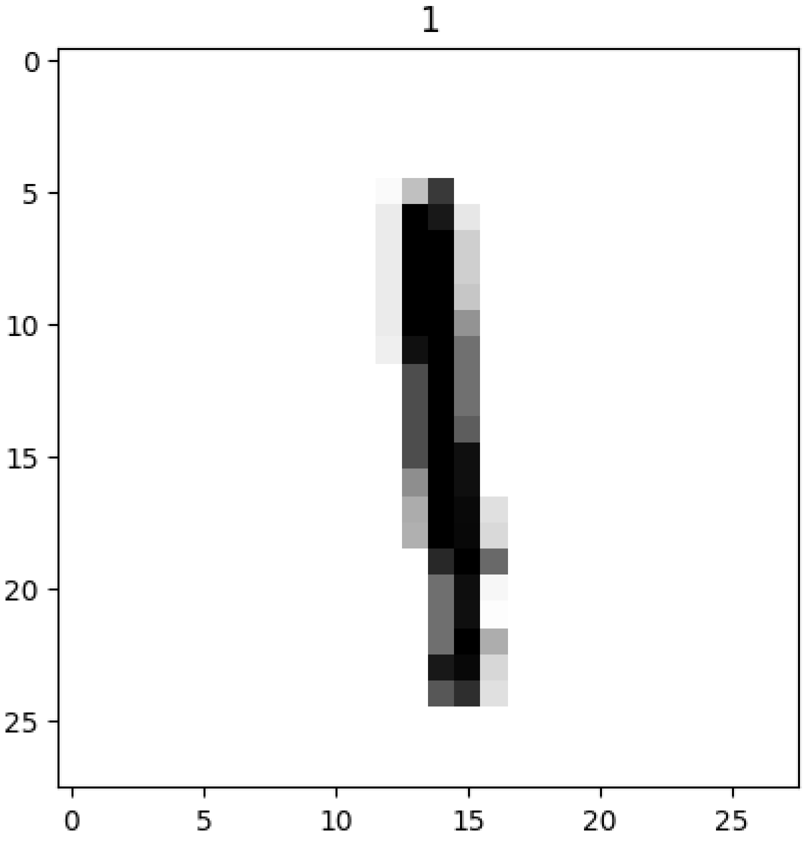

1
0
5
10
15
20
25
0
5
10
15
20
25

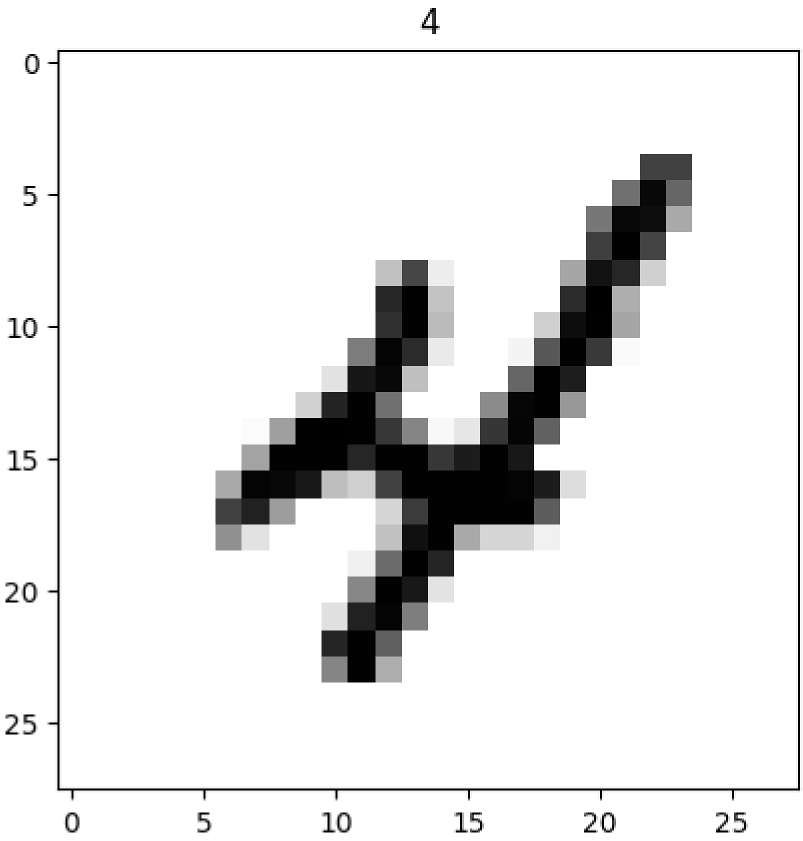

9
0
5
10
15
20
25
0
5
10
15
20
25


2
0
5
10
15
20
25
0
5
10
15
20
25


1
0
5
10
15
20
25
0
5
10
15
20
25


3
0
5
10
15
20
25
0
5
10
15
20
25


1
0
5
10
15
20
25
0
5
10
15
20
25



In [ ]:
```
from os import XATTR_SIZE_MAX
# Preprocessing the images

# Normalizing the images to [0,1] scale
X_train = X_train.astype(np.float32)/255
X_test = X_test.astype(np.float32)/255

# Reshape or expand the dimensations of the images to (28,28,1)
X_train = np.expand_dims(X_train, -1)
X_test = np.expand_dims(X_test, -1)


# Converting classes to one hot vectors
y_train = keras.utils.to_categorical(y_train)
y_test = keras.utils.to_categorical(y_test)
```

In [ ]:
```
X_train.shape
```

Out[ ]:
```
(60000, 28, 28, 1, 1)
```

In [ ]:
```
# Model building

model = Sequential()

model.add(Conv2D(32, (3,3), input_shape = (28,28,1), activation= 'relu'))
model.add(MaxPool2D((2,2)))

model.add(Conv2D(64, (3,3), activation= 'relu'))
model.add(MaxPool2D((2,2)))

model.add(Flatten())

model.add(Dropout(0.25))

model.add(Dense(10, activation= "softmax"))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.p
y:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer.
When using Sequential models, prefer using an `Input(shape)` object as the first l
ayer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

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

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 26, 26, 32)     │           320 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 13, 13, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 11, 11, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 5, 5, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 1600)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 1600)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 10)             │        16,010 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 34,826 (136.04 KB)
 Trainable params: 34,826 (136.04 KB)
 Non-trainable params: 0 (0.00 B)
```

In [ ]:
```
model.compile(optimizer= 'adam', loss = keras.losses.categorical_crossentropy, metr
```

In [ ]:
```
# Callbacks

from keras.callbacks import EarlyStopping, ModelCheckpoint

# EarlyStopping

es = EarlyStopping(monitor='val_acc', min_delta=0.01, patience= 4, verbose= 1)

# Model check point

mc = ModelCheckpoint("./bestmodel.h5", monitor= "val_acc", verbose= 1, save_best_or

cb = [es,mc]
```

In [ ]:
```
# Model training

his = model.fit(X_train, y_train, epochs = 5, validation_split= 0.3)
```

```
Epoch 1/5
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 38s 29ms/step - accuracy: 0.9480 - loss: 0.1698 - v
al_accuracy: 0.9632 - val_loss: 0.1235
Epoch 2/5
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 38s 29ms/step - accuracy: 0.9529 - loss: 0.1524 - v
al_accuracy: 0.9663 - val_loss: 0.1154
Epoch 3/5
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 38s 29ms/step - accuracy: 0.9586 - loss: 0.1420 - v
al_accuracy: 0.9682 - val_loss: 0.1082
Epoch 4/5
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 40s 29ms/step - accuracy: 0.9590 - loss: 0.1381 - v
al_accuracy: 0.9686 - val_loss: 0.1069
Epoch 5/5
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 38s 29ms/step - accuracy: 0.9599 - loss: 0.1334 - v
al_accuracy: 0.9711 - val_loss: 0.0993
```

In [ ]:

```
model_S = keras.models.load_model('my_bestmodel.h5')
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be bu
ilt. `model.compile_metrics` will be empty until you train or evaluate the model.
```

In [ ]:

```
score = model_S.evaluate(X_test, y_test)

print(f"My model accuracy is {score[1]}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9700 - loss: 0.0975
My model accuracy is 0.9750000238418579
```

In [ ]: