```
from google.colab import files
files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle (1).json
{'kaggle.json': b'{"username":"dikshabhati","key":"84f76c6a9c76c3c8d591f12b28ba6927"}'}
```

```
#create a new file and move the kaggle file into it
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!ls ~/.kaggle
```

```
kaggle.json
```

```
!kaggle competitions download -c digit-recognizer
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (serv
test.csv.zip: Skipping, found more recently modified local copy (use --force to force do
sample_submission.csv: Skipping, found more recently modified local copy (use --force to
train.csv.zip: Skipping, found more recently modified local copy (use --force to force d
```

```
!unzip -q "../content/train.csv"
```

```
[../content/train.csv]
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
replace train.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
!unzip -q "../content/test.csv"
```

```
[../content/test.csv]
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
replace test.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
import pandas as pd
train_df=pd.read_csv("../content/train.csv")
test_df=pd.read_csv("../content/test.csv")
```

```
train_df.shape
```

```
(42000, 785)
```

```
test_df.shape
```

```
(28000, 784)
```

```
X_train = train_df.drop(labels = ["label"],axis = 1)
Y_train = train_df["label"]
```

```
X_train.shape
```

```
(42000, 784)
```

```
Y_train
```

```
0        1
1        0
2        1
3        4
4        0
        ..
41995    0
41996    1
41997    7
41998    6
41999    9
Name: label, Length: 42000, dtype: int64
```

```
train_df.dtypes
```

```
label       int64
pixel0      int64
pixel1      int64
pixel2      int64
pixel3      int64
            ...
pixel779    int64
pixel780    int64
pixel781    int64
pixel782    int64
pixel783    int64
Length: 785, dtype: object
```

```
sum(train_df.isnull().sum())
```

```
0
```

```
sum(test_df.isnull().sum())
```

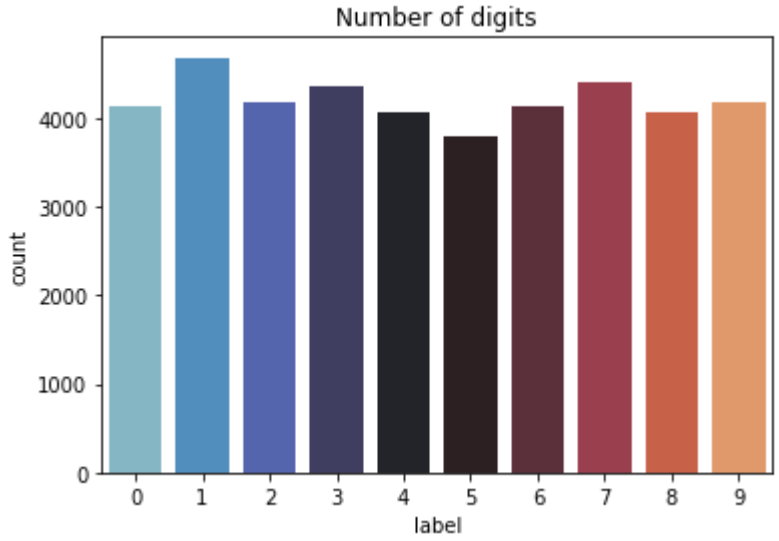
```
0
```

```
Y_train.value_counts()
```

```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(Y_train,palette='icefire')
plt.title("Number of digits")
```

```
Text(0.5, 1.0, 'Number of digits')
```




```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
X_train=X_train.values.reshape(X_train.shape[0],28,28,1)
X_train.shape
```

```
(42000, 28, 28, 1)
```

```
test_df=test_df.values.reshape(test_df.shape[0],28,28,1)
test df.shape
```

```
(28000, 28, 28, 1)
```

```
from keras.utils.np_utils import to_categorical
Y_train=to_categorical(Y_train,num_classes=10)
```

```
Y_train
```

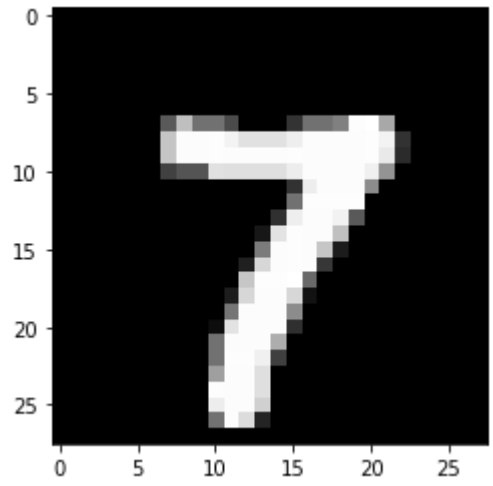
```
array([[0., 1., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)
```

```
X_train = X_train / 255.0
test_df = test_df / 255.0
```

```
from sklearn.model_selection import train_test_split
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1, random_s
```

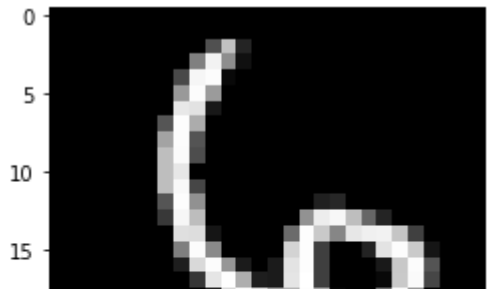
```
plt.imshow(X_train[0][:,:,0],cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7f7cb8424160>
```

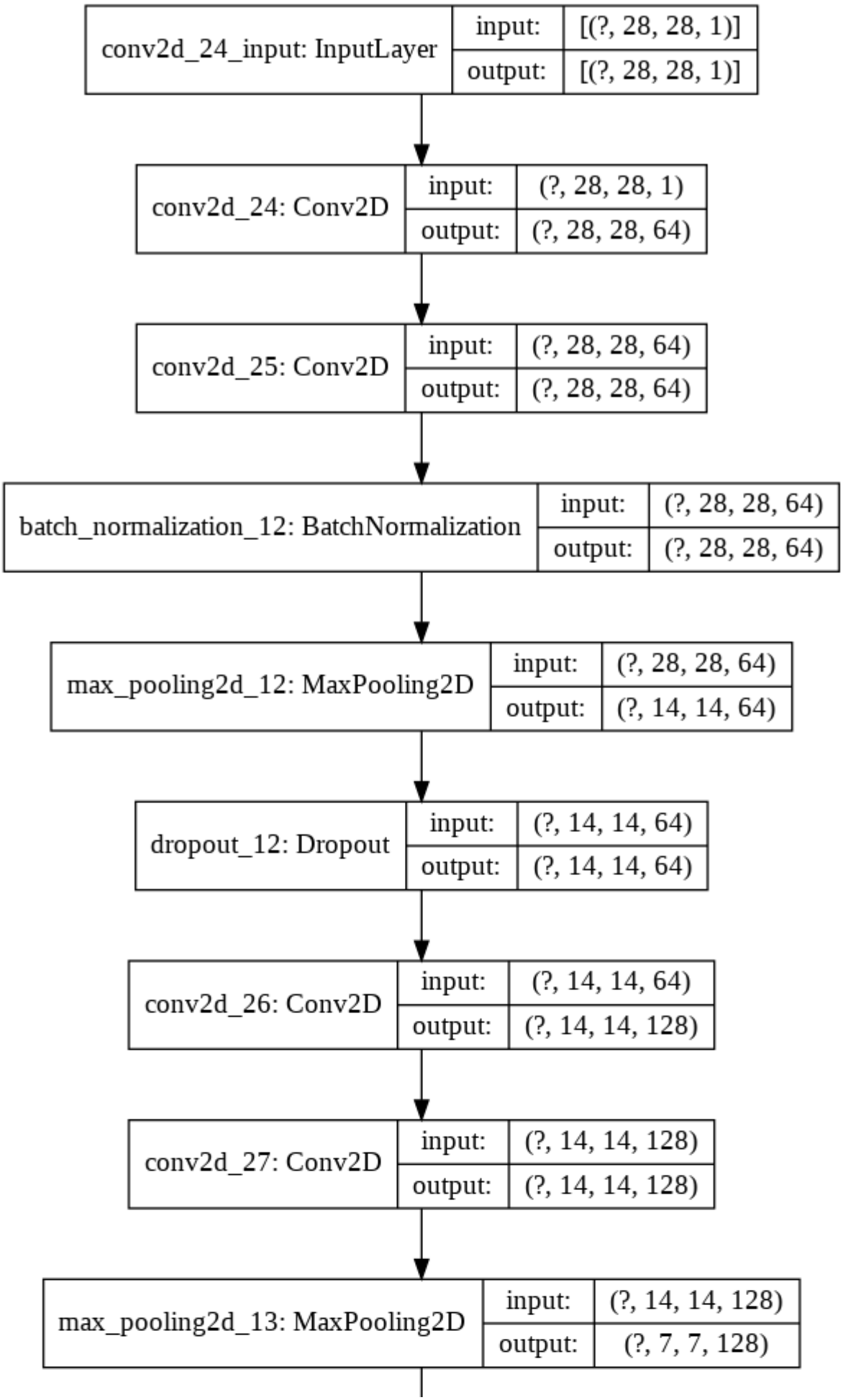


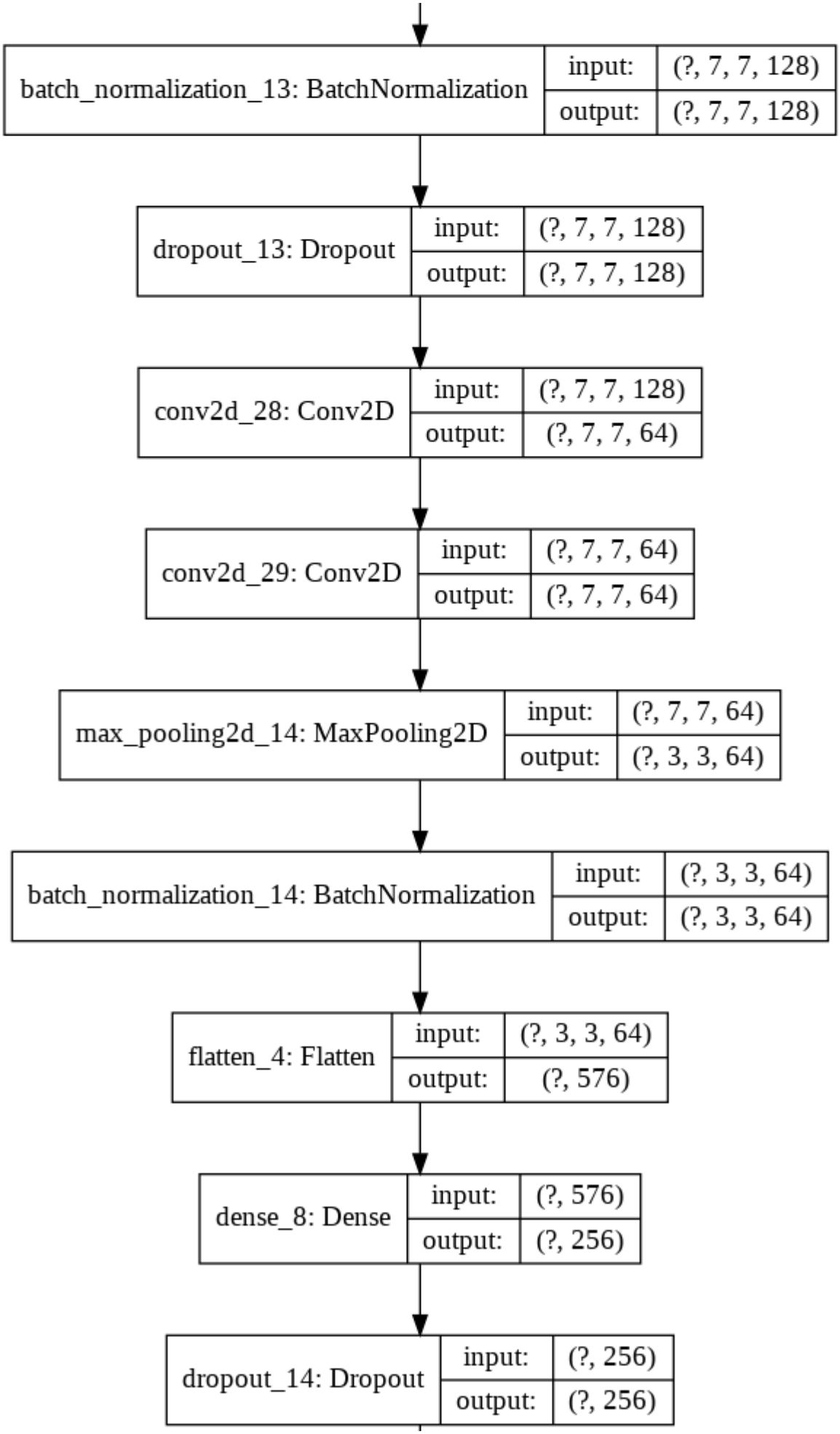

```
plt.imshow(X_train[4][:,:,0],cmap="gray")
```

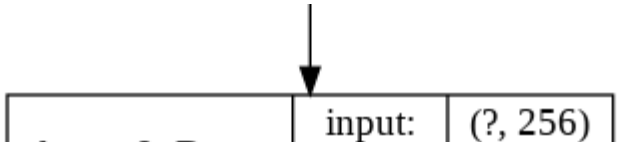
```
<matplotlib.image.AxesImage at 0x7f7cb8e76fd0>
```




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

model=Sequential()

model.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu', input
model.add(Conv2D(filters = 64, kernel_size = (5,5),padding = 'Same',activation ='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))


model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))


from tensorflow import keras
adam=keras.optimizers.Adam(learning_rate=0.001,epsilon=1e-08,beta_1=0.9,beta_2=0.999)


model.compile(optimizer = adam , loss = "categorical_crossentropy", metrics=["accuracy"])


from keras.utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```


conv2d_24_input: InputLayer
input: [(?, 28, 28, 1)]
output: [(?, 28, 28, 1)]
conv2d_24: Conv2D
input: (?, 28, 28, 1)
output: (?, 28, 28, 64)
conv2d_25: Conv2D
input: (?, 28, 28, 64)
output: (?, 28, 28, 64)
batch_normalization_12: BatchNormalization
input: (?, 28, 28, 64)
output: (?, 28, 28, 64)
max_pooling2d_12: MaxPooling2D
input: (?, 28, 28, 64)
output: (?, 14, 14, 64)
dropout_12: Dropout
input: (?, 14, 14, 64)
output: (?, 14, 14, 64)
conv2d_26: Conv2D
input: (?, 14, 14, 64)
output: (?, 14, 14, 128)
conv2d_27: Conv2D
input: (?, 14, 14, 128)
output: (?, 14, 14, 128)
max_pooling2d_13: MaxPooling2D
input: (?, 14, 14, 128)
output: (?, 7, 7, 128)


batch_normalization_13: BatchNormalization
input: (?, 7, 7, 128)
output: (?, 7, 7, 128)
dropout_13: Dropout
input: (?, 7, 7, 128)
output: (?, 7, 7, 128)
conv2d_28: Conv2D
input: (?, 7, 7, 128)
output: (?, 7, 7, 64)
conv2d_29: Conv2D
input: (?, 7, 7, 64)
output: (?, 7, 7, 64)
max_pooling2d_14: MaxPooling2D
input: (?, 7, 7, 64)
output: (?, 3, 3, 64)
batch_normalization_14: BatchNormalization
input: (?, 3, 3, 64)
output: (?, 3, 3, 64)
flatten_4: Flatten
input: (?, 3, 3, 64)
output: (?, 576)
dense_8: Dense
input: (?, 576)
output: (?, 256)
dropout_14: Dropout
input: (?, 256)
output: (?, 256)




```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 28, 28, 64)        1664
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 28, 28, 64)        102464
_________________________________________________________________
batch_normalization_12 (Batc (None, 28, 28, 64)        256
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_12 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 14, 14, 128)       73856
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
batch_normalization_13 (Batc (None, 7, 7, 128)         512
_________________________________________________________________
dropout_13 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 7, 7, 64)          73792
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 3, 3, 64)          0
_________________________________________________________________
batch_normalization_14 (Batc (None, 3, 3, 64)          256
_________________________________________________________________
flatten_4 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               147712
_________________________________________________________________
dropout_14 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                2570
=================================================================
Total params: 587,594
Trainable params: 587,082
Non-trainable params: 512
_________________________________________________________________
```

```
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
data_aug = tf.keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
```

```
data_aug = keras.preprocessing.image.ImageDataGenerator(featurewise_center=False,
                                                         samplewise_center=False,
                                                         featurewise_std_normalization=Fals
                                                         samplewise_std_normalization=False
                                                         zca_whitening=False,
                                                         rotation_range=10,
                                                         zoom_range=0.1,
                                                         width_shift_range=0.1,
                                                         height_shift_range=0.1,
                                                         horizontal_flip=False,
                                                         vertical_flip=False)


data_aug.fit(X_train)


epochs=10
batch_size=20


from keras.optimizers import RMSprop
from sklearn.metrics import confusion_matrix
from keras.callbacks import ReduceLROnPlateau


learning_rate_reduction=ReduceLROnPlateau(monitor='val_acc',patience=1,verbose=2,factor=0.6,m


hist = model.fit_generator(data_aug.flow(X_train,Y_train,batch_size=batch_size),
                 steps_per_epoch=X_train.shape[0]//batch_size,
                 epochs=epochs,
                 validation_data=(X_val,Y_val),
                 validation_steps=10000//batch_size,
                 callbacks=[learning_rate_reduction])
```

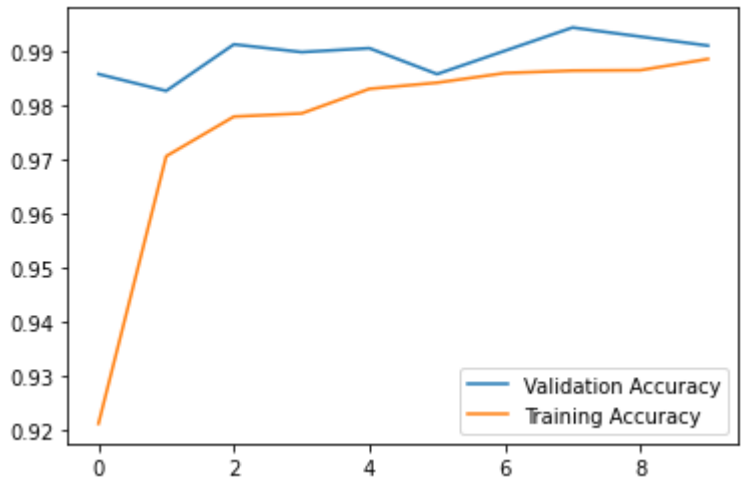
```
Epoch 1/10
1890/1890 [==============================] - ETA: 0s - loss: 0.2824 - accuracy: 0.9126WA
WARNING:tensorflow:Reduce LR on plateau conditioned on metric `val_acc` which is not ava
1890/1890 [==============================] - 794s 420ms/step - loss: 0.2824 - accuracy:
Epoch 2/10
1890/1890 [==============================] - ETA: 0s - loss: 0.1216 - accuracy: 0.9656WA
1890/1890 [==============================] - 766s 405ms/step - loss: 0.1216 - accuracy:
Epoch 3/10
1890/1890 [==============================] - ETA: 0s - loss: 0.0942 - accuracy: 0.9731WA
1890/1890 [==============================] - 766s 405ms/step - loss: 0.0942 - accuracy:
Epoch 4/10
1890/1890 [==============================] - ETA: 0s - loss: 0.0823 - accuracy: 0.9764WA
1890/1890 [==============================] - 766s 405ms/step - loss: 0.0823 - accuracy:
Epoch 5/10
1890/1890 [==============================] - ETA: 0s - loss: 0.0705 - accuracy: 0.9801WA
1890/1890 [==============================] - 764s 404ms/step - loss: 0.0705 - accuracy:
Epoch 6/10
 667/1890 [=========>....................] - ETA: 8:23 - loss: 0.0603 - accuracy: 0.9830
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
print(hist.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```

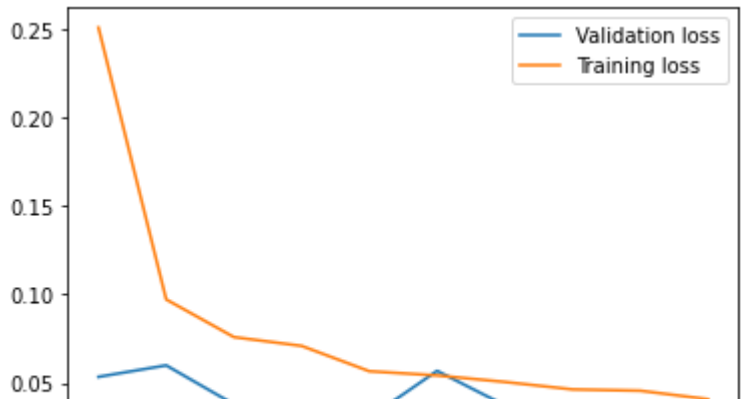
```
plt.plot(hist.history["val_accuracy"],label="Validation Accuracy")
plt.plot(hist.history["accuracy"],label="Training Accuracy")
plt.legend()
plt.plot()
```

```
[]
```




```
plt.plot(hist.history["val_loss"],label="Validation loss")
plt.plot(hist.history["loss"],label="Training loss")
plt.legend()
plt.plot()
```

[]




```
Y_pred=model.predict(test_df)
```

```
Y_pred.shape
```

```
(28000, 10)
```

```
Y_pred[0]
```

```
array([2.3740344e-13, 5.3405846e-11, 1.0000000e+00, 1.9546807e-08,
       1.9335039e-12, 5.3650389e-14, 3.8102445e-13, 1.3427028e-09,
       4.2735740e-10, 5.5917705e-12], dtype=float32)
```

```
import numpy as np
results=np.argmax(Y_pred,axis=1)
results=pd.Series(results,name='Label')
```

```
submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)
submission.to_csv('submission_datarecognizer1.csv',index=False)
```

```
i=9
print("Image label is: ", results[i])
plt.imshow(test_df[i][:,:,0])
```

```
Image label is:  3
<matplotlib.image.AxesImage at 0x7f7cc4cae860>
```