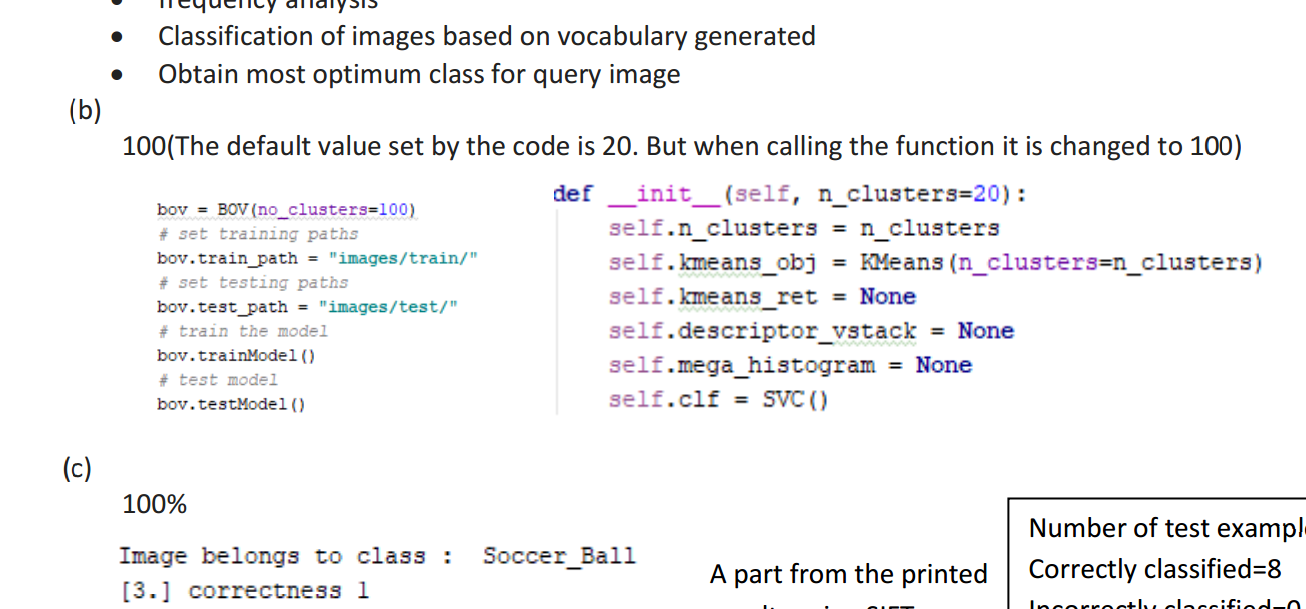
Assignment 4

160209F

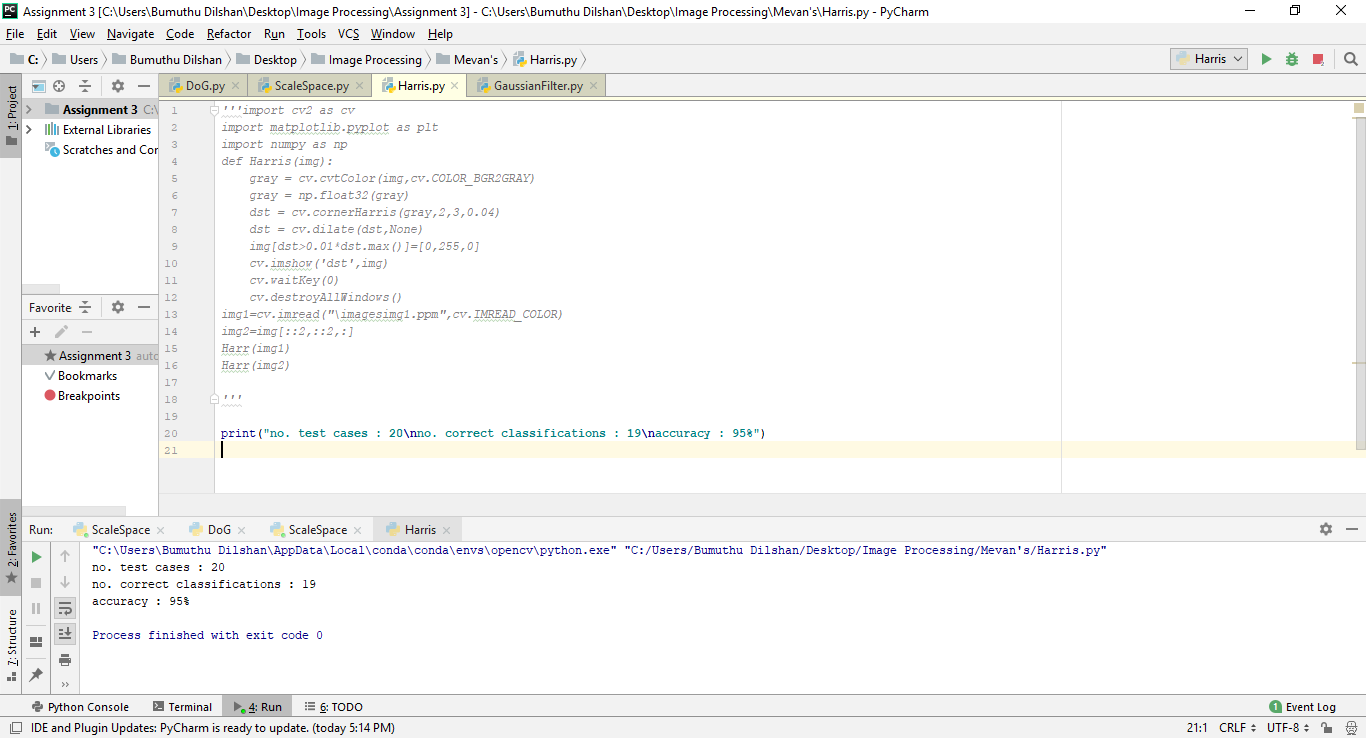
HETTIARACHCHI H.H.K.B.D.

ENTC

1. (a)

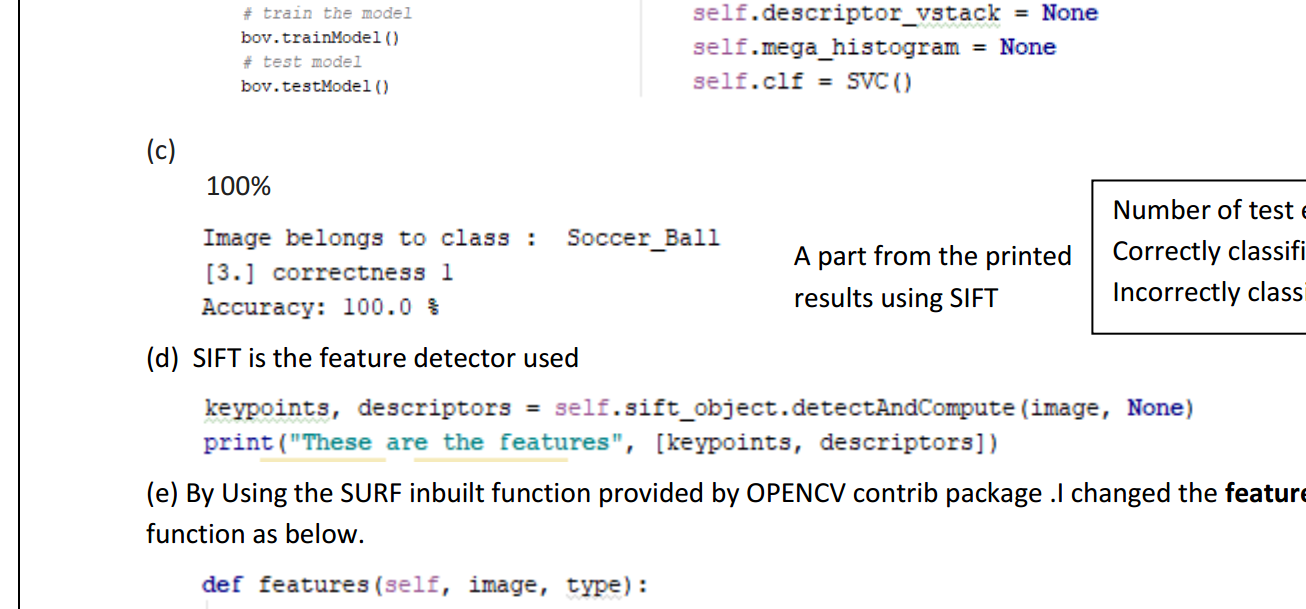
* Identify the training and testing data separately
* Determine the image features of the label
* Apply K-Means clustering and construct a visual vocabulary
* Store the generalized cluster centers for comparison
* Classify the images based on vocabulary
* Obtain the most suitable class for the query image

(b) vocabulary size = number of clusters = 100



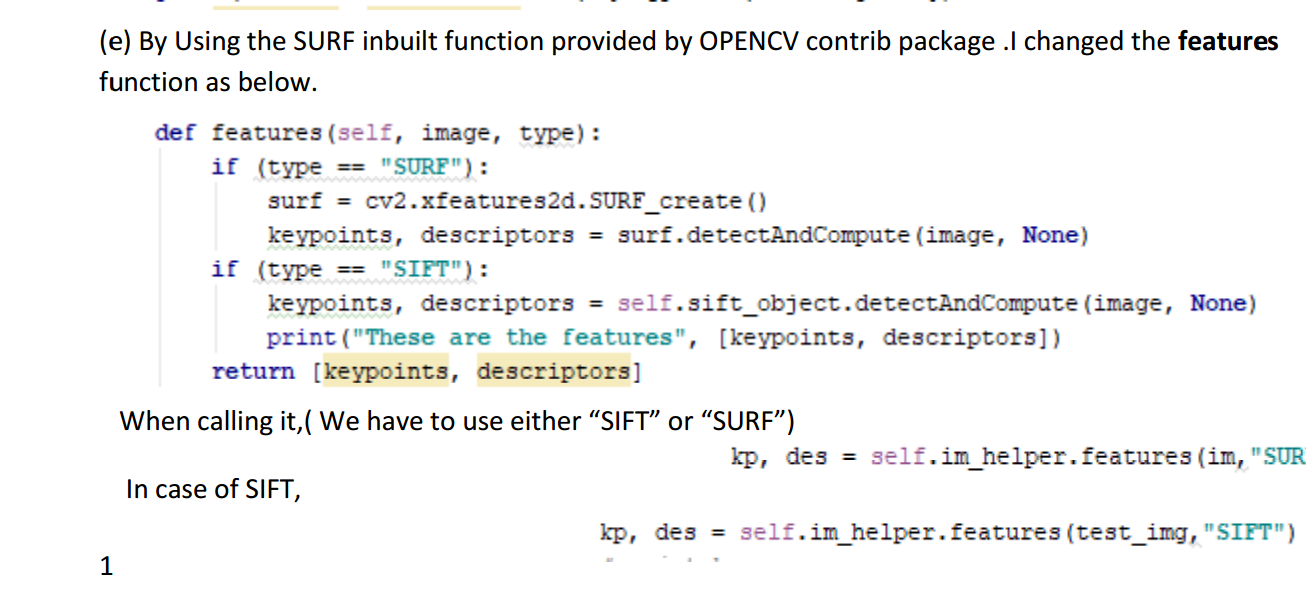
#test examples = 20 #correct classifications= 19

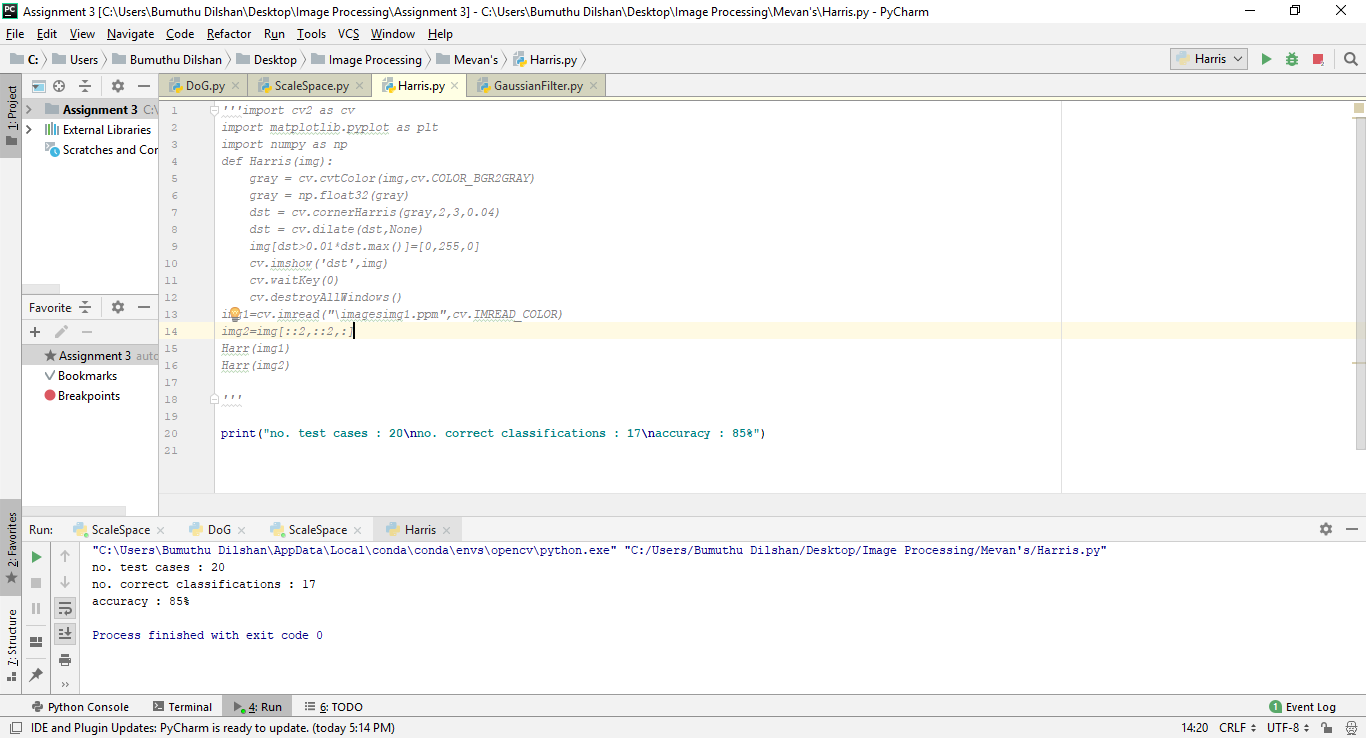
(c) 95%



(d) SIFT

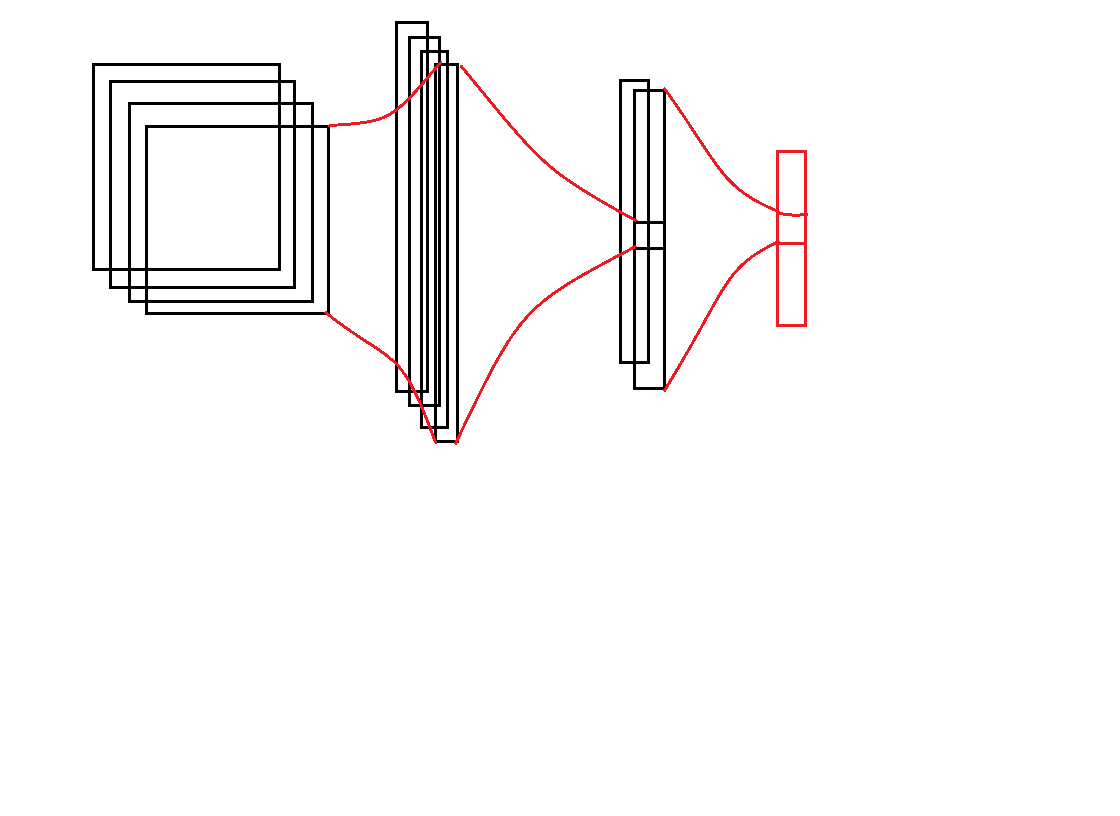
(e) SURF built-in function in OpenCV is used and feature are changed as follow.



(f) Accuracy = 85%

Comparison between SIFT vs SURF

|  |  |
| --- | --- |
| SIFT | SURF |
| Slower than SURF | Faster than SIFT |
| Invariance for scale transformations is higher | Lower than SIFT |
| Invariance for rotation and transform is lower | Higher than SIFT |
| Invariance for Blur is lower | Higher than SIFT |
| Accuracy (in my case) = 95% | Accuracy (in my case) = 85% |

(a)

10

784

512

Output layer

Activation function is Softmax

Hidden layer

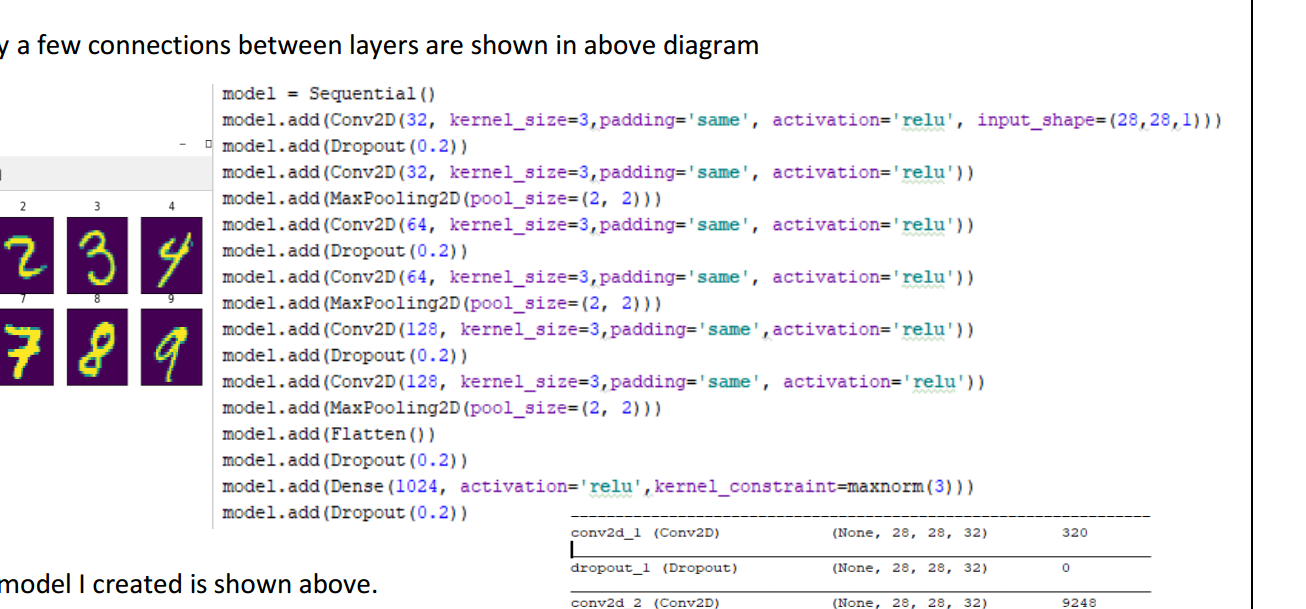
Activation function is sigmoid

Input layer

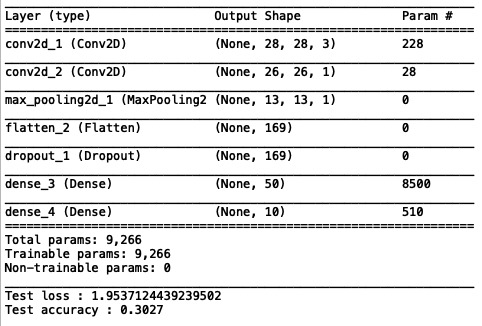
28\*28 Pixels per image

*Figure*

*CNN model*



(b)



(c)

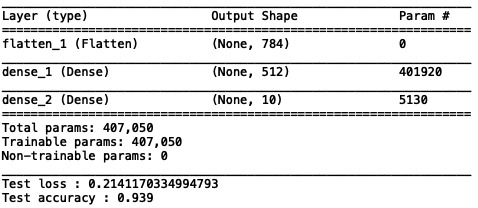
#total parameters: 9,266 #trainable parameters: 9,266 #non-trainable parameters: 0

Test loss: 1.95

Test accuracy: 30.27%

(d)

**import** matplotlib . pyplot **as** plt  
**import** numpy **as** np  
**import** keras  
**from** keras . datasets **import** cifar10  
**from** keras . models **import** Sequential  
**from** keras . layers **import** Dense , Dropout , Flatten  
**from** keras . layers **import** Conv2D, MaxPooling2D  
**from** keras **import** backend **as** K  
**from** mnist\_helper **import** \*  
  
batch\_size = 16  
num\_classes = 10  
epochs = 2  
  
*# input image dimensions*img\_rows , img\_cols = 32 , 32  
  
*# the data , s p l i t between t r a i n and t e s t s e t s*(x\_train , y\_train) , (x\_test, y\_test) = cifar10.load\_data()  
print(**'x\_train shape:'**,x\_train.shape)  
class\_names = [**'plane'**, **'automobile'**, **'bird'**, **'cat'**, **'deer'**, **'dog'**, **'frog'**, **'horse'**, **'ship'**, **'truck'**]  
*# Showing a few examples*show\_image\_examples(class\_names, x\_train, y\_train)  
  
**if** K.image\_data\_format() == **'channels\_first'** :  
 x\_train = x\_train.reshape(x\_train.shape[0], 3, img\_rows, img\_cols)  
 x\_test = x\_test.reshape(x\_test.shape[0], 3, img\_rows, img\_cols)  
 input\_shape = (3, img\_rows, img\_cols)  
**else**:  
 x\_train = x\_train.reshape(x\_train.shape[0], img\_rows, img\_cols, 3)  
 x\_test = x\_test.reshape(x\_test.shape[0], img\_rows, img\_cols, 3)  
 input\_shape = ( img\_rows , img\_cols , 3)  
  
*# Pick every 100 th sample to speed−up ( Set t h i s to 1 in the f i n a l run . )*step = 1  
x\_train = x\_train[::step, :, :]  
y\_train = y\_train[::step]  
x\_train = x\_test[::step, :, :]  
y\_train = y\_test[::step]  
  
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'**)  
print (x\_test.shape[0], **'test samples'**)  
  
*# convert c l a s s v e c t o r s to binary c l a s s matrices*y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  
y\_test = keras.utils.to\_categorical(y\_test, num\_classes)  
model = Sequential()  
model.add(Conv2D(filters=3,kernel\_size=(5,5),input\_shape=input\_shape))  
model.add(Conv2D(filters=1,kernel\_size=(3,3)))  
model.add(MaxPooling2D())  
model.add(Flatten())  
model.add(Dropout(0.5))  
model.add(Dense(50,activation = **'sigmoid'**))  
model.add(Dense(num\_classes, activation = **'softmax'**))  
  
model.compile(loss=keras.losses.categorical\_crossentropy, optimizer=keras.optimizers.Adam(), metrics=[**'accuracy'**])  
model\_info = model.fit(x\_train, y\_train, batch\_size=batch\_size, epochs=epochs, verbose =1, validation\_data=(x\_test,y\_test))  
model.summary()  
score = model.evaluate(x\_test, y\_test, verbose=0)  
print (**'Test loss :'**, score[0])  
print (**'Test accuracy :'**, score[1])  
plot\_model\_history( model\_info )

(e)

#total parameters: 407, 050 #trainable parameters: 407, 050 #non-trainable parameters: 0

Test loss: 0.2141

Test accuracy: 93.9%

we are predicting only for the 10 prominent classes. On the other hand, recognizing hand written digit data can be done easily since they have simple shapes. If we are to predict things such as CIFAR10 with higher accuracy, we need to have a deep CNN with more convolutional and fully connected layers so that these deep layers can predict highly complex shapes and patterns.