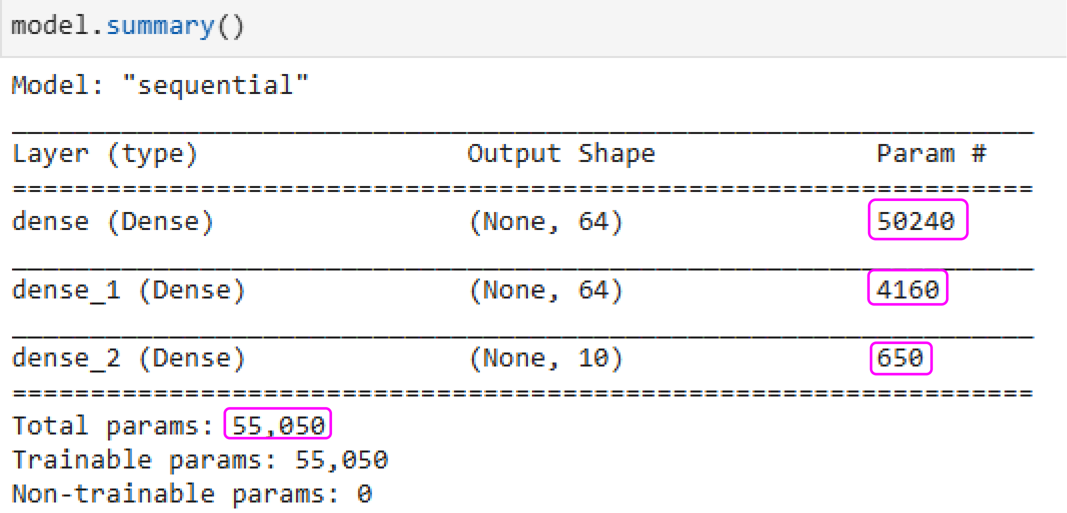
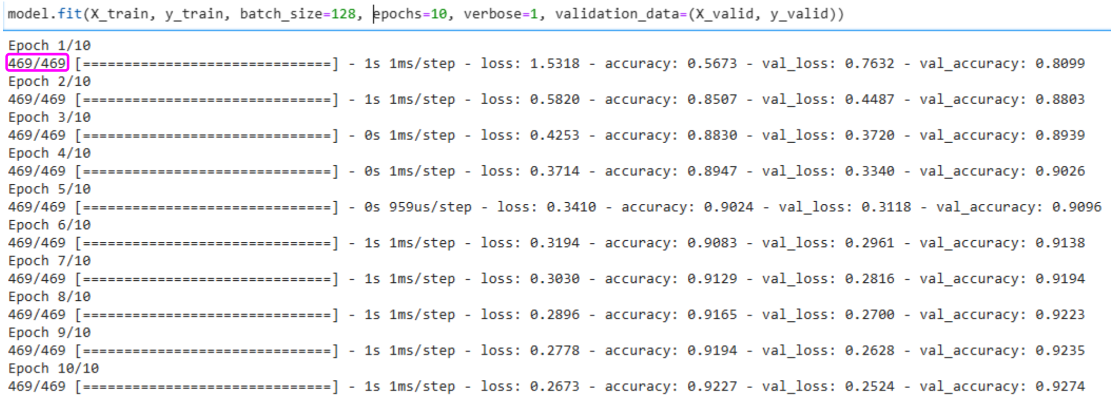
1. Given the file intermediate_net_in_keras.ipynb, modify the file and make it executable. Then, change the learning rate as 0.01 and the number of epochs as 10.
2. Based on the executable file of intermediate_net_in_keras.ipynb, we show the summary of the model. Please give the formula to calculate the number of parameters. There are four numbers you need to give the formulas. For example, how to get 50240 for the line of dense (Dense).



```
model.summary()
```

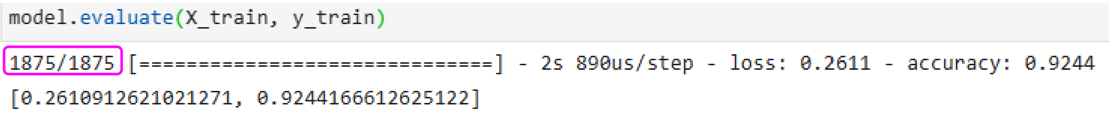
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                50240
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 55,050
Trainable params: 55,050
Non-trainable params: 0
```

3. Based on the executable file of intermediate_net_in_keras.ipynb, add two lines of codes: model.evaluate(X_train, y_train) and model.evaluate(X_valid, y_valid). Please give the formula to calculate the number of iterations. There are three numbers you need to give the formulas. For example, how to get 469 for model.fit.



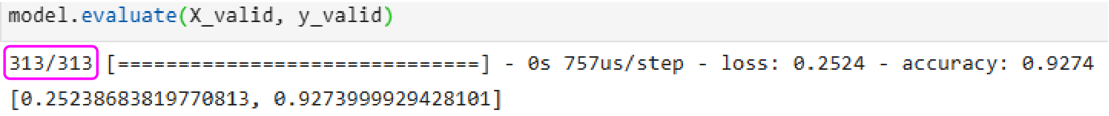
```
model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_valid, y_valid))
```

```
Epoch 1/10
469/469 [==============================] - 1s 1ms/step - loss: 1.5318 - accuracy: 0.5673 - val_loss: 0.7632 - val_accuracy: 0.8099
Epoch 2/10
469/469 [==============================] - 1s 1ms/step - loss: 0.5820 - accuracy: 0.8507 - val_loss: 0.4487 - val_accuracy: 0.8803
Epoch 3/10
469/469 [==============================] - 0s 1ms/step - loss: 0.4253 - accuracy: 0.8830 - val_loss: 0.3720 - val_accuracy: 0.8939
Epoch 4/10
469/469 [==============================] - 0s 1ms/step - loss: 0.3714 - accuracy: 0.8947 - val_loss: 0.3340 - val_accuracy: 0.9026
Epoch 5/10
469/469 [==============================] - 0s 959us/step - loss: 0.3410 - accuracy: 0.9024 - val_loss: 0.3118 - val_accuracy: 0.9096
Epoch 6/10
469/469 [==============================] - 1s 1ms/step - loss: 0.3194 - accuracy: 0.9083 - val_loss: 0.2961 - val_accuracy: 0.9138
Epoch 7/10
469/469 [==============================] - 1s 1ms/step - loss: 0.3030 - accuracy: 0.9129 - val_loss: 0.2816 - val_accuracy: 0.9194
Epoch 8/10
469/469 [==============================] - 1s 1ms/step - loss: 0.2896 - accuracy: 0.9165 - val_loss: 0.2700 - val_accuracy: 0.9223
Epoch 9/10
469/469 [==============================] - 1s 1ms/step - loss: 0.2778 - accuracy: 0.9194 - val_loss: 0.2628 - val_accuracy: 0.9235
Epoch 10/10
469/469 [==============================] - 1s 1ms/step - loss: 0.2673 - accuracy: 0.9227 - val_loss: 0.2524 - val_accuracy: 0.9274
```



```
model.evaluate(X_train, y_train)
```

```
1875/1875 [==============================] - 2s 890us/step - loss: 0.2611 - accuracy: 0.9244
[0.2610912621021271, 0.9244166612625122]
```

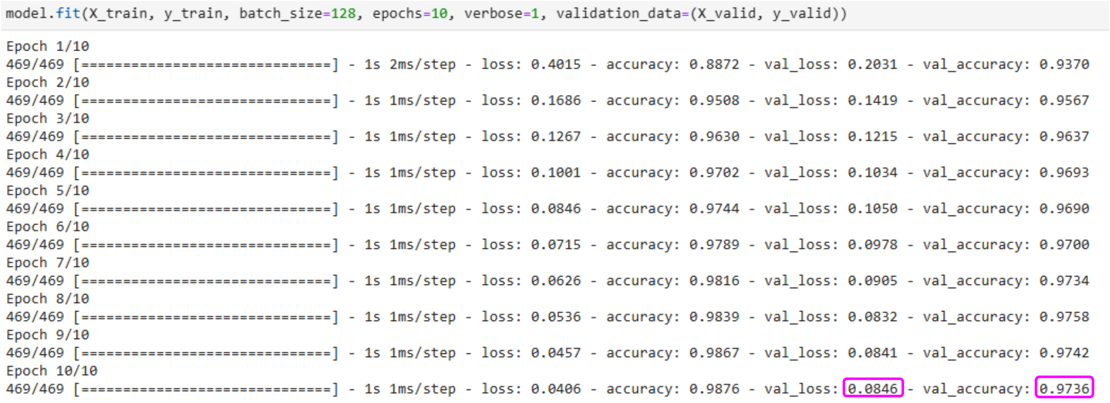


```
model.evaluate(X_valid, y_valid)
```

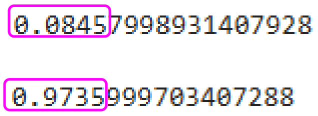
```
313/313 [==============================] - 0s 757us/step - loss: 0.2524 - accuracy: 0.9274
[0.25238683819770813, 0.9273999929428101]
```

4. Based on the executable file of intermediate_net_in_keras.ipynb, change the optimizer as Adam and set the learning rate as 0.001. Then, run the model to see

whether the accuracy of the optimizer Adam is better than that of the optimizer SGD. Add the line of code scores_valid=model.evaluate(X_valid, y_valid) and try to take out the loss and accuracy as follows:

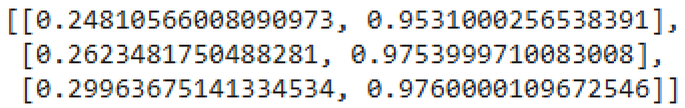


```
model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_valid, y_valid))

Epoch 1/10
469/469 [==============================] - 1s 2ms/step - loss: 0.4015 - accuracy: 0.8872 - val_loss: 0.2031 - val_accuracy: 0.9370
Epoch 2/10
469/469 [==============================] - 1s 1ms/step - loss: 0.1686 - accuracy: 0.9508 - val_loss: 0.1419 - val_accuracy: 0.9567
Epoch 3/10
469/469 [==============================] - 1s 1ms/step - loss: 0.1267 - accuracy: 0.9630 - val_loss: 0.1215 - val_accuracy: 0.9637
Epoch 4/10
469/469 [==============================] - 1s 1ms/step - loss: 0.1001 - accuracy: 0.9702 - val_loss: 0.1034 - val_accuracy: 0.9693
Epoch 5/10
469/469 [==============================] - 1s 1ms/step - loss: 0.0846 - accuracy: 0.9744 - val_loss: 0.1050 - val_accuracy: 0.9690
Epoch 6/10
469/469 [==============================] - 1s 1ms/step - loss: 0.0715 - accuracy: 0.9789 - val_loss: 0.0978 - val_accuracy: 0.9700
Epoch 7/10
469/469 [==============================] - 1s 1ms/step - loss: 0.0626 - accuracy: 0.9816 - val_loss: 0.0905 - val_accuracy: 0.9734
Epoch 8/10
469/469 [==============================] - 1s 1ms/step - loss: 0.0536 - accuracy: 0.9839 - val_loss: 0.0832 - val_accuracy: 0.9758
Epoch 9/10
469/469 [==============================] - 1s 1ms/step - loss: 0.0457 - accuracy: 0.9867 - val_loss: 0.0841 - val_accuracy: 0.9742
Epoch 10/10
469/469 [==============================] - 1s 1ms/step - loss: 0.0406 - accuracy: 0.9876 - val_loss: 0.0846 - val_accuracy: 0.9736
```



```
0.08457998931407928

0.9735999703407288
```

5. Based on the executable file of intermediate_net_in_keras.ipynb, change the optimizer as RMSprop, add the line of code scores_valid=model.evaluate(X_valid, y_valid), and then run the model with the learning rate as 0.01, 0.001, and 0.0001. Finally, collect all three groups of loss and accuracy together as follows:



```
[[0.24810566008090973, 0.9531000256538391],
 [0.2623481750488281, 0.9753999710083008],
 [0.29963675141334534, 0.9760000109672546]]
```

The first list is from the learning rate as 0.01, the second from 0.001, and the third from 0.0001.

Note that your answer generally is different from the result mentioned above.