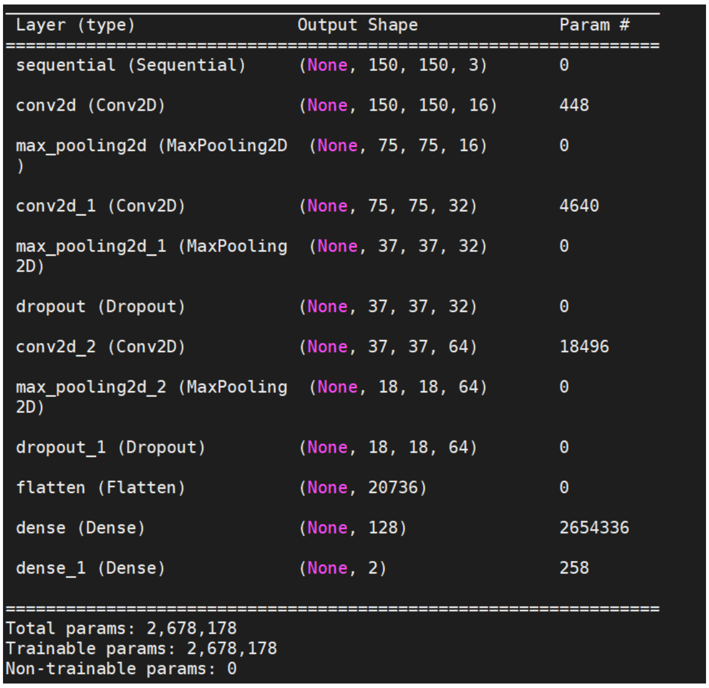
= = Assignment 3 = =

107062313 黃寶萱

- **Model architecture and hyperparameters**

  The model is constructed with 3 2D convolutional layers with activate function 'relu' and the number of 3x3 filters is (16, 32, 64) respectively. Each convolutional layer is followed with MaxPooling layer. To avoid overfitting, I set the 2 Dropout layers. One of Dropout layer is connected to 2nd MaxPooling layer, the other one is connected to 3rd MaxPooling layer. Then, set Flatten layer to change dimension. The last two layers of the model are Dense layer, containing 128 neurons with activation function 'relu' and 2 neurons with activation function 'softmax' respectively.

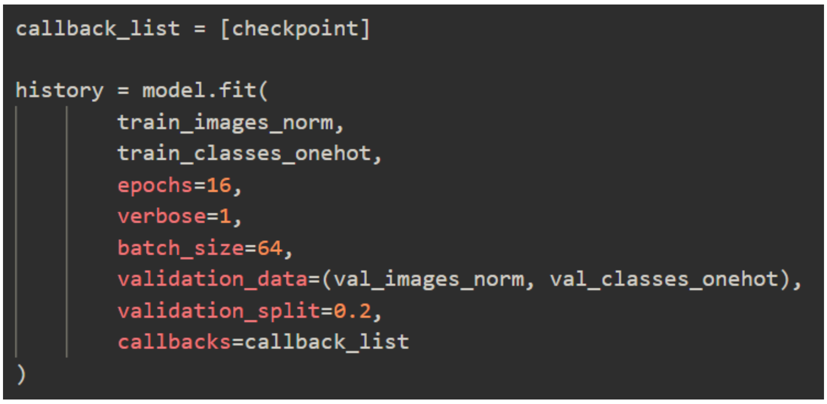


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 150, 150, 3)       0

 conv2d (Conv2D)             (None, 150, 150, 16)      448

 max_pooling2d (MaxPooling2D  (None, 75, 75, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 dropout (Dropout)           (None, 37, 37, 32)        0

 conv2d_2 (Conv2D)           (None, 37, 37, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 18, 18, 64)        0

 flatten (Flatten)           (None, 20736)             0

 dense (Dense)               (None, 128)               2654336

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 2,678,178
Trainable params: 2,678,178
Non-trainable params: 0
```

  To improve the accuracy when evaluate the model's performance, I add the small sequential model to do data augmentation. It will process flip, rotation and zoom randomly with input image to make input data more complex. And when I evaluate the test set, the accuracy is effectively increased from 70 to over 80.

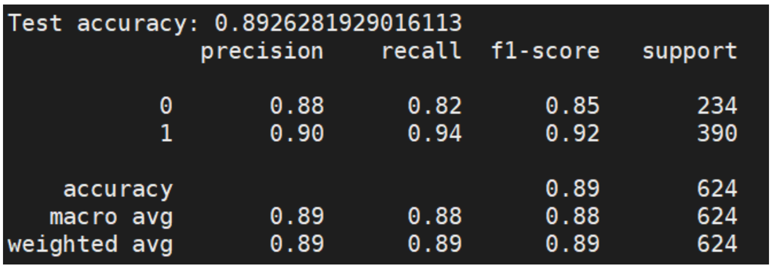
- **Model training arguments setting**

  For all image data, I process normalization first and also do onehot encoding with label data. I set epochs = 16, batch_size=64 and add validation data to evaluate while training. The callbacks is for checkpoint to save history model.

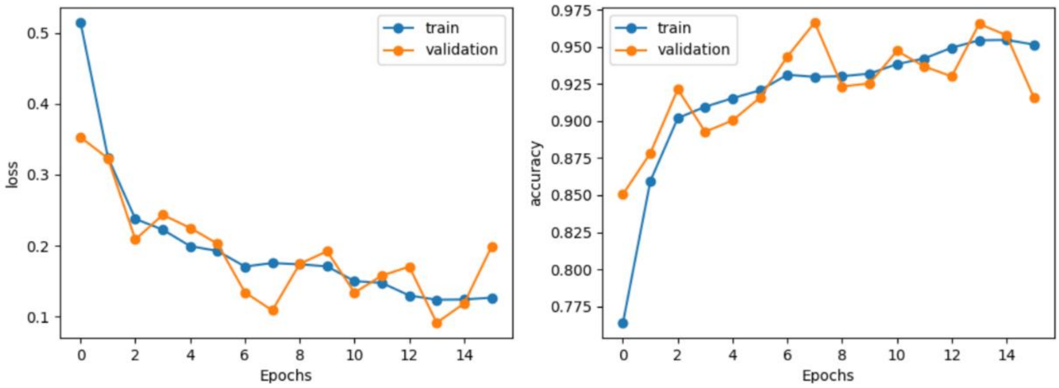


```
callback_list = [checkpoint]

history = model.fit(
        train_images_norm,
        train_classes_onehot,
        epochs=16,
        verbose=1,
        batch_size=64,
        validation_data=(val_images_norm, val_classes_onehot),
        validation_split=0.2,
        callbacks=callback_list
)
```

- **Confusion matrix of the classification performance**



```
Test accuracy: 0.8926281929016113
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       234
           1       0.90      0.94      0.92       390

    accuracy                           0.89       624
   macro avg       0.89      0.88      0.88       624
weighted avg       0.89      0.89      0.89       624
```



- **Deal with overfit problem**

To avoid overfit problem, I don't use early stop strategy since the model only run 16 epochs while training. I have tried to use early stop strategy to improve evaluate performance; however, the accuracy was not increase. Finally, I add data augmentation strategy to increase the difficulty of input data and also add 2 Dropout layers to reduce model complexity. After several tests, this strategy really works for the model, the evaluate accuracy is around 82 to 90.