```
!pip  install  tensorflow==2.1.0
```

資源 ×

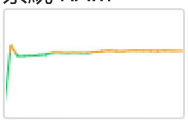
需要更多記憶體和磁碟空間嗎?

升級至 Colab Pro

Python 3 Google Compute Engine 後端 (GPU)

目前顯示自 凌晨2:16以來的資源

系統 RAM



GPU RAM

磁碟

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.
Collecting tensorflow==2.1.0
  Downloading tensorflow-2.1.0-cp37-cp37m-manylinux2010_x86_64.whl
     |████████████████████████████████| 421.8 MB 26 kB/s
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/pyth
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/pyth
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lil
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/py
Collecting keras-applications>=1.0.8
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 7.7 MB/s
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/p
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/py
Collecting tensorboard<2.2.0,>=2.1.0
  Downloading tensorboard-2.1.1-py3-none-any.whl (3.8 MB)
     |████████████████████████████████| 3.8 MB 63.4 MB/s
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/pyth
Collecting scipy==1.4.1
  Downloading scipy-1.4.1-cp37-cp37m-manylinux1_x86_64.whl (26.1 MB
     |████████████████████████████████| 26.1 MB 1.4 MB/s
Collecting tensorflow-estimator<2.2.0,>=2.1.0rc0
  Downloading tensorflow_estimator-2.1.0-py2.py3-none-any.whl (448 k
     |████████████████████████████████| 448 kB 67.2 MB/s
Requirement already satisfied: keras-preprocessing>=1.1.0 in /usr/lo
```

```
!pip  install  keras==2.3.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/col
Collecting keras==2.3.1
  Downloading Keras-2.3.1-py2.py3-none-any.whl (377 kB)
     |████████████████████████████████| 37
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/li
Requirement already satisfied: keras-applications>=1.0.6 in /usr/local/lib
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/di
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.7/dis
Requirement already satisfied: cached-property in /usr/local/lib/python3.7
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: keras 2.8.0
    Uninstalling keras-2.8.0:
      Successfully uninstalled keras-2.8.0
Successfully installed keras-2.3.1
```

```
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/
```

```
!pip  install  h5py==2.10.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/col
Collecting h5py==2.10.0
  Downloading h5py-2.10.0-cp37-cp37m-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.7/dist
Installing collected packages: h5py
  Attempting uninstall: h5py
    Found existing installation: h5py 3.1.0
    Uninstalling h5py-3.1.0:
      Successfully uninstalled h5py-3.1.0
Successfully installed h5py-2.10.0
```

```
Attempting uninstall: scipy
```

```
import  tensorflow  as  tf
print('版本:',tf.__version__)
import  keras
print('版本:',keras.__version__)
```

```
版本: 2.1.0
版本: 2.3.1
Using TensorFlow backend.
```

```
import  h5py
print('版本:',h5py.__version__)
```

```
版本: 2.10.0
```

```
import  numpy  as  np
import  cv2,glob
import  matplotlib.pyplot  as  plt
%matplotlib  inline
```

```
x_real  =  np.load('/content/x_real_fp.npz')['data']
y_real  =  np.load('/content/y_real_fp.npy')
x_easy  =  np.load('/content/x_easy_fp.npz')['data']
y_easy  =  np.load('/content/y_easy_fp.npy')
x_medium  =  np.load('/content/x_medium_fp.npz')['data']
y_medium  =  np.load('/content/y_medium_fp.npy')
x_hard  =  np.load('/content/x_hard_fp.npz')['data']
y_hard  =  np.load('/content/y_hard_fp.npy')
```

```
print(x_real.shape,y_real.shape)
print(x_easy.shape,y_easy.shape)
print(x_medium.shape,y_medium.shape)
print(x_hard.shape,y_hard.shape)
```

```
(6000, 96, 96, 3) (6000, 4)
(17931, 96, 96, 3) (17931, 4)
(17067, 96, 96, 3) (17067, 4)
(14272, 96, 96, 3) (14272, 4)
```

```
from  sklearn.model_selection  import  train_test_split
```

```
x_all_data  =  np.concatenate([x_easy,  x_medium,  x_hard],  axis=0)
label_all_data  =  np.concatenate([y_easy,  y_medium,  y_hard],  axis=0)
print(x_all_data.shape,  label_all_data.shape)
permutation  =  list(np.random.permutation(x_all_data.shape[0]))  #  permutation
x_data  =  x_all_data[permutation][:6000]
label_data  =  label_all_data[permutation][:6000]
x_train,  x_val,  label_train,  label_val  =  train_test_split(x_data,  label_da
print(x_data.shape,  label_data.shape)
print(x_train.shape,  label_train.shape)  #  訓練
print(x_val.shape,  label_val.shape)  #  驗證
```

```
(49270, 96, 96, 3) (49270, 4)
(6000, 96, 96, 3) (6000, 4)
(4800, 96, 96, 3) (4800, 4)
(1200, 96, 96, 3) (1200, 4)
```

```
permutation  =  list(np.random.permutation(x_real.shape[0]))  #  permutation  隨
```

```
x_test  =  x_real[permutation][:1200]
label_test  =  y_real[permutation][:1200]
print(x_test.shape,  label_test.shape)  #  測試
```

```
(1200, 96, 96, 3) (1200, 4)
```

```
from  tensorflow.keras.applications.resnet50  import  ResNet50
from  tensorflow.keras.applications.resnet50  import  preprocess_input
from  tensorflow.keras.models  import  Sequential
from  tensorflow.keras.layers  import  Dense,  Dropout,  GlobalAveragePooling2D
from  tensorflow.keras.utils  import  to_categorical
from  PIL  import  Image
```

```
rs_model  =  ResNet50(include_top=False,  weights="imagenet",input_shape=(120,12
```

```
#train  data

print('調整X_train的圖片尺寸...')
x_train_new  =  np.array([np.asarray(Image.fromarray(x_train[i]).resize((120,  1
                                          for  i  in  range(0,  len(x_trai
x_train_new  =  x_train_new.astype('float32')

print(x_train_new.shape)
```

```
調整X_train的圖片尺寸...
(4800, 120, 120, 3)
```

```
#訓練資料前處理
train_input=preprocess_input(x_train_new)
print('使用  RasNet50  模型預測訓練資料的特徵資料...')
train_features  =  rs_model.predict(train_input,verbose=1)
```

```
使用 RasNet50 模型預測訓練資料的特徵資料...
4800/4800 [==============================] - 11s 2ms/sample
```

```
print(train_features.shape)
```

```
(4800, 4, 4, 2048)
```

```
#val_data

print('調整X_val的圖片尺寸...')
x_val_new  =  np.array([np.asarray(Image.fromarray(x_val[i]).resize((120,  120))
for  i  in  range(0,  len(x_val))])
x_val_new  =  x_val_new.astype('float32')
print(x_val_new.shape)
```

```
調整X_val的圖片尺寸...
(1200, 120, 120, 3)
```

```
#驗證資料前處理

val_input=preprocess_input(x_val_new)
print('使用  RasNet50  模型預測驗證資料的特徵資料...')
val_features  =  rs_model.predict(val_input,verbose=1)
```

```
使用 RasNet50 模型預測驗證資料的特徵資料...
1200/1200 [==============================] - 2s 2ms/sample
```

```
print(val_features.shape)
```

```
(1200, 4, 4, 2048)
```

```
#test  data

print('調整x_test的圖片尺寸...')
x_test_new  =  np.array([np.asarray(Image.fromarray(x_test[i]).resize((120,  120)
for  i  in  range(0,  len(x_test))])
x_test_new  =  x_test_new.astype('float32')
print(x_test_new.shape)
```

```
調整x_test的圖片尺寸...
(1200, 120, 120, 3)
```

```
#測測試資料前處理

test_input=preprocess_input(x_test_new)
print('使用  RasNet50  模型預測測試資料的特徵資料...')
test_features  =  rs_model.predict(test_input,verbose=1)
```

```
使用 RasNet50 模型預測測試資料的特徵資料...
1200/1200 [==============================] - 2s 1ms/sample
```

```
print(test_features.shape)
```

```
(1200, 4, 4, 2048)
```

```
#標籤正規化

print('訓練資料:')
id_label_train  =  to_categorical(label_train[:,0]-1)
print('身分',id_label_train.shape)
gender_label_train  =  to_categorical(label_train[:,1])
print('性別',gender_label_train.shape)
LRhand_label_train  =  to_categorical(label_train[:,2])
print('左右手',LRhand_label_train.shape)
finger_label_train  =  to_categorical(label_train[:,3])
print('指頭',finger_label_train.shape)

print('驗證  :')
id_label_val  =  to_categorical(label_val[:,0]-1)
print('身分',id_label_val.shape)
gender_label_val  =  to_categorical(label_val[:,1])
print('性別',gender_label_val.shape)
LRhand_label_val  =  to_categorical(label_val[:,2])
print('左右手',LRhand_label_val.shape)
finger_label_val  =  to_categorical(label_val[:,3])
print('指頭',finger_label_val.shape)

print('測試資料:')
id_label_test  =  to_categorical(label_test[:,0]-1)
print('身分',id_label_test.shape)
gender_label_test  =  to_categorical(label_test[:,1])
```

```
print('性別',gender_label_test.shape)
LRhand_label_test  =  to_categorical(label_test[:,2])
print('左右手',LRhand_label_test.shape)
finger_label_test  =  to_categorical(label_test[:,3])
print('指頭',finger_label_test.shape)
```

```
訓練資料:
身分 (4800, 600)
性別 (4800, 2)
左右手 (4800, 2)
指頭 (4800, 5)
驗證 :
身分 (1200, 600)
性別 (1200, 2)
左右手 (1200, 2)
指頭 (1200, 5)
測試資料:
身分 (1200, 600)
性別 (1200, 2)
左右手 (1200, 2)
指頭 (1200, 5)
```

```
l()
ragePooling2D(input_shape=train_features.shape[1:]))
.5))
,   activation='softmax'))

'categorical_crossentropy',  optimizer='adam',metrics=['accuracy'])

it(train_features,id_label_train,  validation_data=(val_features,  id_label_val),
                    epochs=100,  batch_size=32,  verbose=2)

")
 model.evaluate(test_features,id_label_test,verbose=1)
確度  =  {:.2f}".format(accuracy))
```

```
Epoch 75/100
4800/4800 - 1s - loss: 0.1356 - accuracy: 0.9581 - val_loss: 9.5164 - v
Epoch 76/100
4800/4800 - 1s - loss: 0.1197 - accuracy: 0.9660 - val_loss: 9.4921 - v
Epoch 77/100
4800/4800 - 1s - loss: 0.1287 - accuracy: 0.9571 - val_loss: 9.5193 - v
Epoch 78/100
4800/4800 - 1s - loss: 0.1223 - accuracy: 0.9617 - val_loss: 9.8619 - v
Epoch 79/100
4800/4800 - 1s - loss: 0.1432 - accuracy: 0.9575 - val_loss: 9.9154 - v
Epoch 80/100
4800/4800 - 1s - loss: 0.1295 - accuracy: 0.9600 - val_loss: 9.8928 - v
Epoch 81/100
4800/4800 - 1s - loss: 0.1163 - accuracy: 0.9652 - val_loss: 9.9805 - v
Epoch 82/100
4800/4800 - 1s - loss: 0.1172 - accuracy: 0.9658 - val_loss: 9.6189 - v
Epoch 83/100
4800/4800 - 1s - loss: 0.1300 - accuracy: 0.9640 - val_loss: 9.9552 - v
Epoch 84/100
4800/4800 - 1s - loss: 0.1038 - accuracy: 0.9683 - val_loss: 9.8257 - v
Epoch 85/100
4800/4800 - 1s - loss: 0.1248 - accuracy: 0.9652 - val_loss: 9.7355 - v
Epoch 86/100
4800/4800 - 1s - loss: 0.1097 - accuracy: 0.9673 - val_loss: 9.8249 - v
```

```
Epoch 87/100
4800/4800 - 1s - loss: 0.1220 - accuracy: 0.9631 - val_loss: 9.8714 - v
Epoch 88/100
4800/4800 - 1s - loss: 0.1370 - accuracy: 0.9619 - val_loss: 9.9039 - v
Epoch 89/100
4800/4800 - 1s - loss: 0.1146 - accuracy: 0.9629 - val_loss: 9.8894 - v
Epoch 90/100
4800/4800 - 1s - loss: 0.1210 - accuracy: 0.9629 - val_loss: 10.0087 -
Epoch 91/100
4800/4800 - 1s - loss: 0.1153 - accuracy: 0.9654 - val_loss: 10.0204 -
Epoch 92/100
4800/4800 - 1s - loss: 0.1283 - accuracy: 0.9646 - val_loss: 10.0929 -
Epoch 93/100
4800/4800 - 1s - loss: 0.1427 - accuracy: 0.9592 - val_loss: 10.1163 -
Epoch 94/100
4800/4800 - 1s - loss: 0.0905 - accuracy: 0.9735 - val_loss: 10.0984 -
Epoch 95/100
4800/4800 - 1s - loss: 0.1181 - accuracy: 0.9648 - val_loss: 10.5288 -
Epoch 96/100
4800/4800 - 1s - loss: 0.1073 - accuracy: 0.9652 - val_loss: 10.1346 -
Epoch 97/100
4800/4800 - 1s - loss: 0.1201 - accuracy: 0.9652 - val_loss: 10.1082 -
Epoch 98/100
4800/4800 - 1s - loss: 0.0979 - accuracy: 0.9706 - val_loss: 10.1697 -
Epoch 99/100
4800/4800 - 1s - loss: 0.1223 - accuracy: 0.9677 - val_loss: 10.2680 -
Epoch 100/100
4800/4800 - 1s - loss: 0.1026 - accuracy: 0.9694 - val_loss: 10.3172 -

Testing...
1200/1200 [==============================] - 0s 155us/sample - loss: 8.
測試資料集準確度 = 0.37
```

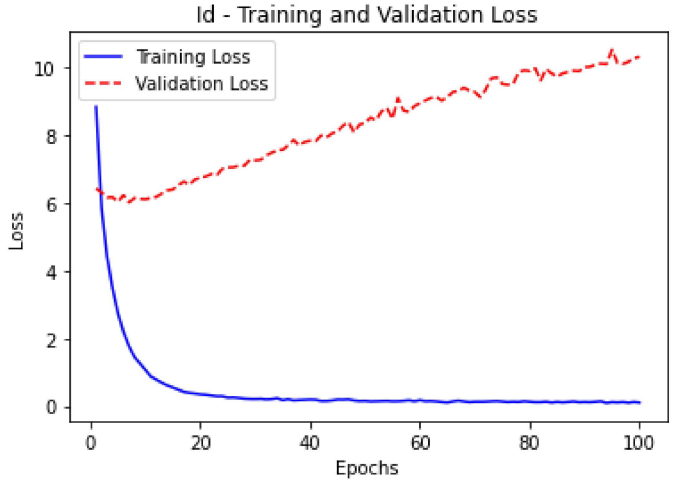
```
model.summary()
```

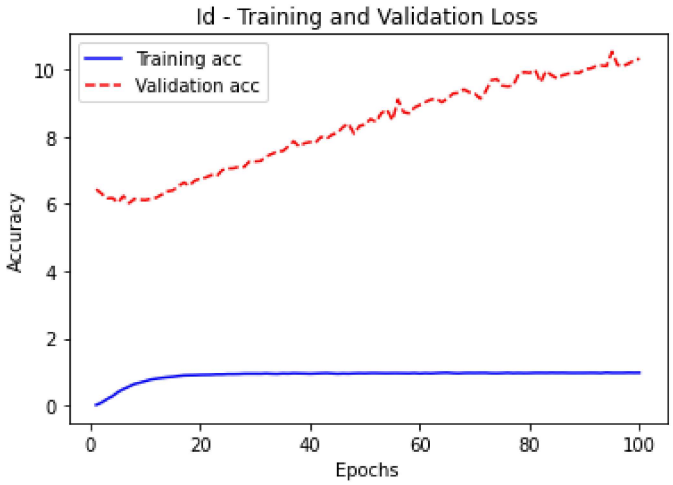
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 600)               1229400
=================================================================
Total params: 1,229,400
Trainable params: 1,229,400
Non-trainable params: 0
_________________________________________________________________
```

```
model.save('/content/resnet50_fpAll_id.h5')
```

```
#  顯示訓練和測試損失
loss  =  history.history['loss']
epochs  =  range(1,  len(loss)  +  1)
val_loss  =  history.history['val_loss']
plt.plot(epochs,  loss,  'b-',  label='Training  Loss')
plt.plot(epochs,  val_loss,  'r--',  label='Validation  Loss')
plt.title('Id  -  Training  and  Validation  Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
plt.show()

acc = history.history['accuracy']
epochs = range(1, len(acc) + 1)
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'b-', label='Training acc')
plt.plot(epochs, val_loss, 'r--', label='Validation acc')
plt.title('Id - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```






```
;2D(input_shape=train_features.shape[1:]))

ı='softmax'))

)ssentropy', optimizer='adam',metrics=['accuracy'])

_features,gender_label_train, validation_data=(val_features, gender_label_val),
        epochs=100, batch_size=32, verbose=2)

.uate(test_features,gender_label_test,verbose=1)
?f}".format(accuracy))
```

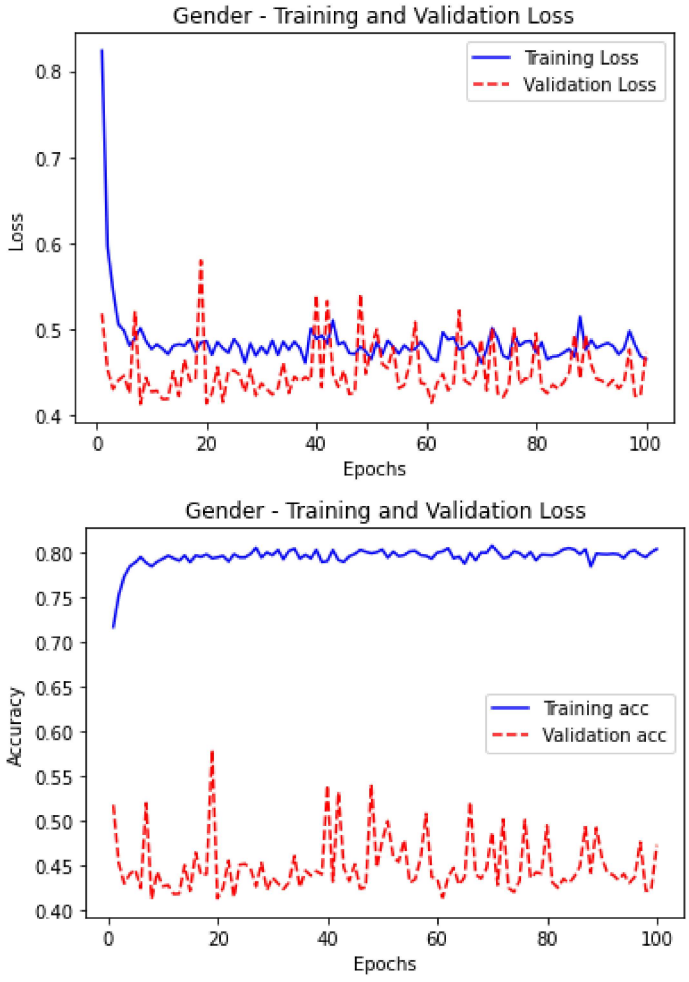
```
4800/4800 - 1s - loss: 0.4649 - accuracy: 0.7992 - val_loss: 0.4324 - v
Epoch 76/100
4800/4800 - 1s - loss: 0.4936 - accuracy: 0.7942 - val_loss: 0.5003 - v
Epoch 77/100
```

```
4800/4800 - 1s - loss: 0.4800 - accuracy: 0.8004 - val_loss: 0.4355 - v
Epoch 78/100
4800/4800 - 1s - loss: 0.4849 - accuracy: 0.7910 - val_loss: 0.4419 - v
Epoch 79/100
4800/4800 - 1s - loss: 0.4857 - accuracy: 0.7975 - val_loss: 0.4407 - v
Epoch 80/100
4800/4800 - 1s - loss: 0.4722 - accuracy: 0.7971 - val_loss: 0.4943 - v
Epoch 81/100
4800/4800 - 1s - loss: 0.4840 - accuracy: 0.7967 - val_loss: 0.4306 - v
Epoch 82/100
4800/4800 - 1s - loss: 0.4642 - accuracy: 0.7994 - val_loss: 0.4247 - v
Epoch 83/100
4800/4800 - 1s - loss: 0.4673 - accuracy: 0.8033 - val_loss: 0.4347 - v
Epoch 84/100
4800/4800 - 1s - loss: 0.4680 - accuracy: 0.8048 - val_loss: 0.4299 - v
Epoch 85/100
4800/4800 - 1s - loss: 0.4721 - accuracy: 0.8031 - val_loss: 0.4374 - v
Epoch 86/100
4800/4800 - 1s - loss: 0.4762 - accuracy: 0.7979 - val_loss: 0.4484 - v
Epoch 87/100
4800/4800 - 1s - loss: 0.4673 - accuracy: 0.8033 - val_loss: 0.4925 - v
Epoch 88/100
4800/4800 - 1s - loss: 0.5139 - accuracy: 0.7842 - val_loss: 0.4414 - v
Epoch 89/100
4800/4800 - 1s - loss: 0.4750 - accuracy: 0.7985 - val_loss: 0.4919 - v
Epoch 90/100
4800/4800 - 1s - loss: 0.4867 - accuracy: 0.7979 - val_loss: 0.4581 - v
Epoch 91/100
4800/4800 - 1s - loss: 0.4771 - accuracy: 0.7977 - val_loss: 0.4414 - v
Epoch 92/100
4800/4800 - 1s - loss: 0.4806 - accuracy: 0.7983 - val_loss: 0.4392 - v
Epoch 93/100
4800/4800 - 1s - loss: 0.4832 - accuracy: 0.7977 - val_loss: 0.4335 - v
Epoch 94/100
4800/4800 - 1s - loss: 0.4797 - accuracy: 0.7935 - val_loss: 0.4402 - v
Epoch 95/100
4800/4800 - 1s - loss: 0.4700 - accuracy: 0.8006 - val_loss: 0.4302 - v
Epoch 96/100
4800/4800 - 1s - loss: 0.4767 - accuracy: 0.8023 - val_loss: 0.4369 - v
Epoch 97/100
4800/4800 - 1s - loss: 0.4973 - accuracy: 0.7973 - val_loss: 0.4757 - v
Epoch 98/100
4800/4800 - 1s - loss: 0.4806 - accuracy: 0.7946 - val_loss: 0.4215 - v
Epoch 99/100
4800/4800 - 1s - loss: 0.4678 - accuracy: 0.8000 - val_loss: 0.4221 - v
Epoch 100/100
4800/4800 - 1s - loss: 0.4641 - accuracy: 0.8037 - val_loss: 0.4725 - v

Testing...


1200/1200 [==============================] - 0s 120us/sample - loss: 0.
測試資料集準確度 = 0.80
```

```
model_g.save('/content/resnet50_fpAll_gneder.h5')
```

```
#  顯示訓練和測試損失
loss  =  history_g.history['loss']
epochs  =  range(1,  len(loss)  +  1)
val_loss  =  history_g.history['val_loss']
plt.plot(epochs,  loss,  'b-',  label='Training  Loss')
plt.plot(epochs,  val_loss,  'r--',  label='Validation  Loss')
plt.title('Gender  -  Training  and  Validation  Loss')
plt.xlabel('Epochs')
```

```
plt.ylabel('Loss')
plt.legend()
plt.show()

acc = history_g.history['accuracy']
epochs = range(1, len(acc) + 1)
val_acc = history_g.history['val_accuracy']
plt.plot(epochs, acc, 'b-', label='Training acc')
plt.plot(epochs, val_loss, 'r--', label='Validation acc')
plt.title('Gender - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
#LRhand model
model_LR = Sequential()
model_LR.add(GlobalAveragePooling2D(input_shape=train_features.shape[1:]))
model_LR.add(Dropout(0.5))
model_LR.add(Dense(2, activation='softmax'))

model_LR.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accurac

history_LR = model_LR.fit(train_features,LRhand_label_train, validation_data=
                                    epochs=100, batch_size=32, verbose=2)
print("\nTesting...")
loss, accuracy = model_LR.evaluate(test_features,LRhand_label_test,verbose=1)
print("測試資料集準確度 = {:.2f}".format(accuracy))
```

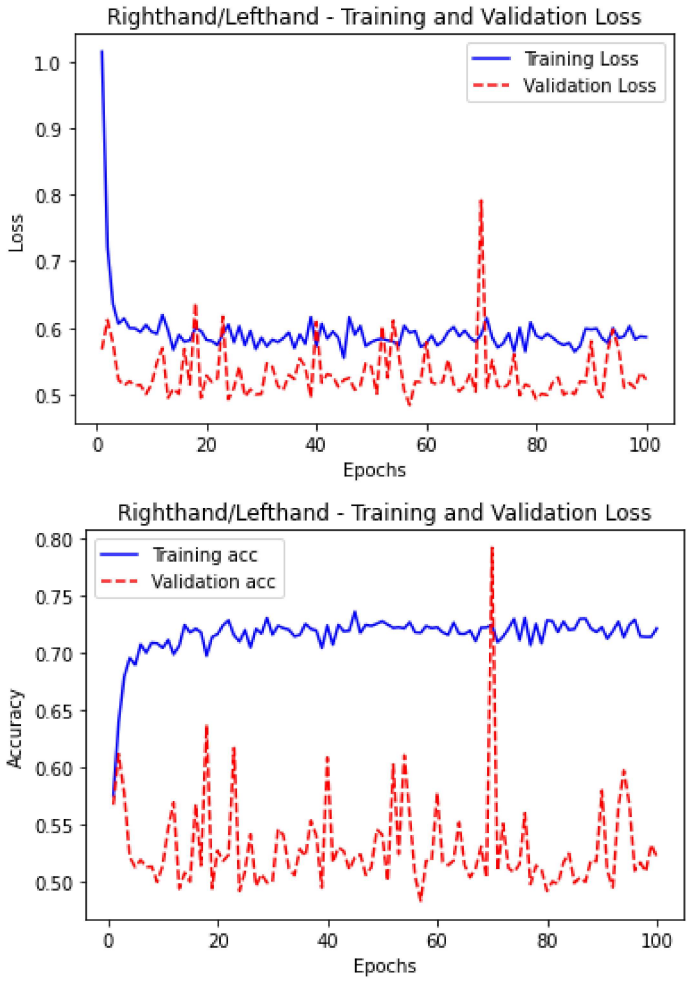
```
Epoch 75/100
4800/4800 - 1s - loss: 0.5919 - accuracy: 0.7108 - val_loss: 0.5160 - v
```

```
Epoch 76/100
4800/4800 - 1s - loss: 0.5644 - accuracy: 0.7304 - val_loss: 0.5604 - v
Epoch 77/100
4800/4800 - 1s - loss: 0.6001 - accuracy: 0.7065 - val_loss: 0.4983 - v
Epoch 78/100
4800/4800 - 1s - loss: 0.5643 - accuracy: 0.7250 - val_loss: 0.5147 - v
Epoch 79/100
4800/4800 - 1s - loss: 0.6080 - accuracy: 0.7081 - val_loss: 0.5112 - v
Epoch 80/100
4800/4800 - 1s - loss: 0.5881 - accuracy: 0.7281 - val_loss: 0.4925 - v
Epoch 81/100
4800/4800 - 1s - loss: 0.5836 - accuracy: 0.7275 - val_loss: 0.5008 - v
Epoch 82/100
4800/4800 - 1s - loss: 0.5907 - accuracy: 0.7179 - val_loss: 0.4988 - v
Epoch 83/100
4800/4800 - 1s - loss: 0.5852 - accuracy: 0.7271 - val_loss: 0.5173 - v
Epoch 84/100
4800/4800 - 1s - loss: 0.5781 - accuracy: 0.7198 - val_loss: 0.5252 - v
Epoch 85/100
4800/4800 - 1s - loss: 0.5739 - accuracy: 0.7204 - val_loss: 0.4994 - v
Epoch 86/100
4800/4800 - 1s - loss: 0.5772 - accuracy: 0.7294 - val_loss: 0.5043 - v
Epoch 87/100
4800/4800 - 1s - loss: 0.5641 - accuracy: 0.7296 - val_loss: 0.5004 - v
Epoch 88/100
4800/4800 - 1s - loss: 0.5726 - accuracy: 0.7210 - val_loss: 0.5194 - v
Epoch 89/100
4800/4800 - 1s - loss: 0.5986 - accuracy: 0.7179 - val_loss: 0.5182 - v
Epoch 90/100
4800/4800 - 1s - loss: 0.5974 - accuracy: 0.7221 - val_loss: 0.5801 - v
Epoch 91/100
4800/4800 - 1s - loss: 0.5988 - accuracy: 0.7121 - val_loss: 0.5098 - v
Epoch 92/100
4800/4800 - 1s - loss: 0.5845 - accuracy: 0.7190 - val_loss: 0.4957 - v
Epoch 93/100
4800/4800 - 1s - loss: 0.5772 - accuracy: 0.7269 - val_loss: 0.5648 - v
Epoch 94/100
4800/4800 - 1s - loss: 0.5995 - accuracy: 0.7133 - val_loss: 0.5975 - v
Epoch 95/100
4800/4800 - 1s - loss: 0.5844 - accuracy: 0.7242 - val_loss: 0.5683 - v
Epoch 96/100
4800/4800 - 1s - loss: 0.5871 - accuracy: 0.7287 - val_loss: 0.5102 - v
Epoch 97/100
4800/4800 - 1s - loss: 0.6023 - accuracy: 0.7142 - val_loss: 0.5172 - v
Epoch 98/100
4800/4800 - 1s - loss: 0.5826 - accuracy: 0.7138 - val_loss: 0.5093 - v
Epoch 99/100
4800/4800 - 1s - loss: 0.5870 - accuracy: 0.7138 - val_loss: 0.5329 - v
Epoch 100/100
4800/4800 - 1s - loss: 0.5860 - accuracy: 0.7210 - val_loss: 0.5222 - v


Testing...
1200/1200 [==============================] - 0s 125us/sample - loss: 0.
測試資料集準確度 = 0.74
```

```
model_LR.save('/content/resnet50_fpAll_LR.h5')
```

```
#  顯示訓練和測試損失
loss  =  history_LR.history['loss']
epochs  =  range(1,  len(loss)  +  1)
val_loss  =  history_LR.history['val_loss']
plt.plot(epochs,  loss,  'b-',  label='Training  Loss')
plt.plot(epochs,  val_loss,  'r--',  label='Validation  Loss')
plt.title('Rightband/Leftband  -  Training  and  Validation  Loss')
```

```
plt.title('Righthand/Lefthand - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

acc = history_LR.history['accuracy']
epochs = range(1, len(acc) + 1)
val_acc = history_LR.history['val_accuracy']
plt.plot(epochs, acc, 'b-', label='Training acc')
plt.plot(epochs, val_loss, 'r--', label='Validation acc')
plt.title('Righthand/Lefthand - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



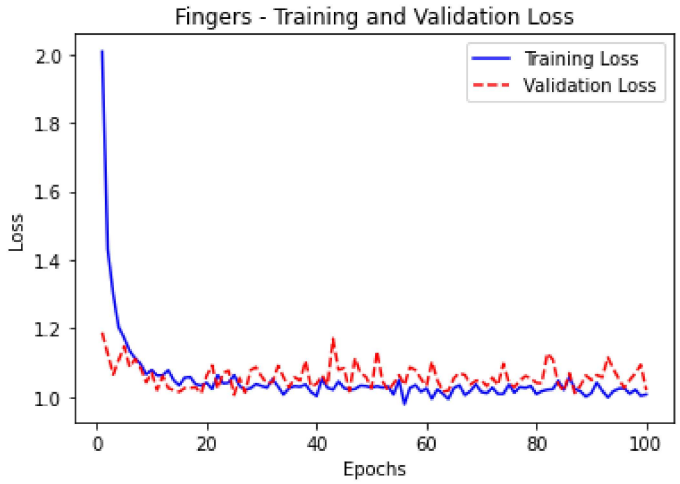

```
#finger model
model_f = Sequential()
model_f.add(GlobalAveragePooling2D(input_shape=train_features.shape[1:]))
model_f.add(Dropout(0.5))
model_f.add(Dense(5, activation='softmax'))

model_f.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['acc

history_f = model_f.fit(train_features,finger_label_train, validation_data=(v
                                    epochs=100, batch_size=32, verbose=2)
print("\nTesting...")
loss, accuracy = model_f.evaluate(test_features,finger_label_test,verbose=1)
print("測試資料集準確度 = {:.2f}".format(accuracy))
```

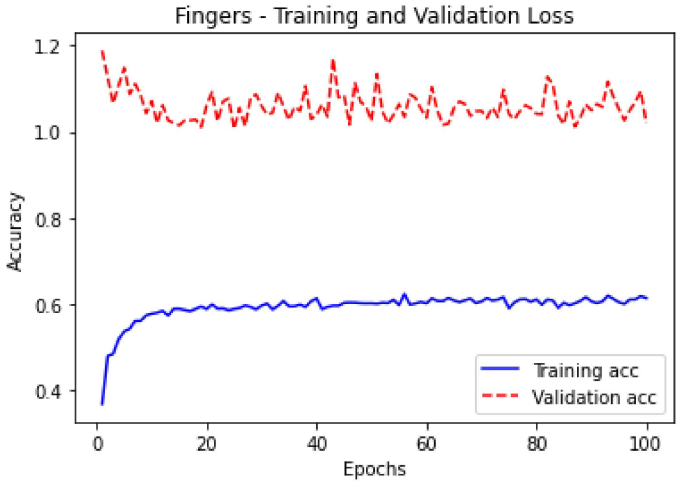
```
Epoch 75/100
4800/4800 - 1s - loss: 1.0367 - accuracy: 0.5902 - val_loss: 1.0408 - v
Epoch 76/100
4800/4800 - 1s - loss: 1.0126 - accuracy: 0.6042 - val_loss: 1.0290 - v
Epoch 77/100
4800/4800 - 1s - loss: 1.0290 - accuracy: 0.6110 - val_loss: 1.0475 - v
Epoch 78/100
4800/4800 - 1s - loss: 1.0267 - accuracy: 0.6119 - val_loss: 1.0612 - v
Epoch 79/100
4800/4800 - 1s - loss: 1.0322 - accuracy: 0.6056 - val_loss: 1.0547 - v
Epoch 80/100
4800/4800 - 1s - loss: 1.0086 - accuracy: 0.6112 - val_loss: 1.0414 - v
Epoch 81/100
4800/4800 - 1s - loss: 1.0172 - accuracy: 0.5987 - val_loss: 1.0405 - v
Epoch 82/100
4800/4800 - 1s - loss: 1.0217 - accuracy: 0.6110 - val_loss: 1.1276 - v
Epoch 83/100
4800/4800 - 1s - loss: 1.0230 - accuracy: 0.6087 - val_loss: 1.1092 - v
Epoch 84/100
4800/4800 - 1s - loss: 1.0480 - accuracy: 0.5913 - val_loss: 1.0373 - v
Epoch 85/100
4800/4800 - 1s - loss: 1.0210 - accuracy: 0.6033 - val_loss: 1.0176 - v
Epoch 86/100
4800/4800 - 1s - loss: 1.0589 - accuracy: 0.5975 - val_loss: 1.0698 - v
Epoch 87/100
4800/4800 - 1s - loss: 1.0234 - accuracy: 0.6019 - val_loss: 1.0117 - v
Epoch 88/100
4800/4800 - 1s - loss: 1.0170 - accuracy: 0.6077 - val_loss: 1.0328 - v
Epoch 89/100
4800/4800 - 1s - loss: 1.0013 - accuracy: 0.6162 - val_loss: 1.0629 - v
Epoch 90/100
4800/4800 - 1s - loss: 1.0117 - accuracy: 0.6067 - val_loss: 1.0495 - v
Epoch 91/100
4800/4800 - 1s - loss: 1.0424 - accuracy: 0.6025 - val_loss: 1.0643 - v
Epoch 92/100
4800/4800 - 1s - loss: 1.0184 - accuracy: 0.6062 - val_loss: 1.0573 - v
Epoch 93/100
4800/4800 - 1s - loss: 0.9989 - accuracy: 0.6196 - val_loss: 1.1162 - v
Epoch 94/100
4800/4800 - 1s - loss: 1.0170 - accuracy: 0.6125 - val_loss: 1.0820 - v
Epoch 95/100
4800/4800 - 1s - loss: 1.0246 - accuracy: 0.6046 - val_loss: 1.0527 - v
Epoch 96/100
4800/4800 - 1s - loss: 1.0254 - accuracy: 0.6004 - val_loss: 1.0265 - v
Epoch 97/100
4800/4800 - 1s - loss: 1.0098 - accuracy: 0.6108 - val_loss: 1.0494 - v
Epoch 98/100
4800/4800 - 1s - loss: 1.0215 - accuracy: 0.6110 - val_loss: 1.0689 - v
Epoch 99/100
4800/4800 - 1s - loss: 1.0038 - accuracy: 0.6185 - val_loss: 1.0943 - v
Epoch 100/100
4800/4800 - 1s - loss: 1.0077 - accuracy: 0.6142 - val_loss: 1.0198 - v


Testing...
1200/1200 [==============================] - 0s 117us/sample - loss: 0.
測試資料集準確度 = 0.64
```

```
model_f.save('/content/resnet50_fpAll_finger.h5')
```

```
#  顯示訓練和測試損失
loss  =  history_f.history['loss']
epochs  =  range(1,  len(loss)  +  1)
val_loss  =  history_f.history['val_loss']
```

```
plt.plot(epochs, loss, 'b-', label='Training Loss')
plt.plot(epochs, val_loss, 'r--', label='Validation Loss')
plt.title('Fingers - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

acc = history_f.history['accuracy']
epochs = range(1, len(acc) + 1)
val_acc = history_f.history['val_accuracy']
plt.plot(epochs, acc, 'b-', label='Training acc')
plt.plot(epochs, val_loss, 'r--', label='Validation acc')
plt.title('Fingers - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```





管理工作階段

變更執行階段類型

✓ 0 秒　完成時間：凌晨3:05