ARTIST CLASSIFICATION

資管二 A_蘇湘婷_109403009

Colab :
https://colab.research.google.com/drive/1iVbSzjlEPBqyGrySHygkA_k_QuWW0ldX?usp=sharing

Test Acc:

```
#  讀入測試資料並評估模型
test_ds  =  make_dataset(test_dir)
test_ds  =  test_ds.batch(batch_size)
score  =  model_best.evaluate(test_ds)
print("Test  loss:",  score[0])
print("Test  accuracy:",  score[1])
```

```
27/27 [==============================] - 3s 16ms/step - loss: 2.5216 - accuracy: 0.4982
Test loss: 2.521562337875366
Test accuracy: 0.4982036054134369
```

程式撰寫過程:
最終版本的 code 的 epochs 跟 learning_rate 設定總共寫了四個版本，最高的 Test_Acc=49.8%

1、
epochs = 50, learning_rate=0.0002
epochs = 100, learning_rate=0.0001

```
[ ]  #  讀入測試資料並評估模型
     test_ds  =  make_dataset(test_dir)
     test_ds  =  test_ds.batch(batch_size)
     score  =  model.evaluate(test_ds)
     print("Test  loss:",  score[0])
     print("Test  accuracy:",  score[1])
```

```
27/27 [==============================] - 3s 12ms/step - loss: 2.4819 - accuracy: 0.4754
Test loss: 2.4818930625915527
Test accuracy: 0.4754491150379181
```

2、
epochs = 50, learning_rate=0.0002
epochs = 100, learning_rate=0.0001
epochs = 120, learning_rate=0.00005

```
[ ]  #  讀入測試資料並評估模型
     test_ds  =  make_dataset(test_dir)
     test_ds  =  test_ds.batch(batch_size)
     score  =  model.evaluate(test_ds)
     print("Test  loss:",  score[0])
     print("Test  accuracy:",  score[1])
```

```
27/27 [==============================] - 3s 12ms/step - loss: 2.6977 - accuracy: 0.4587
Test loss: 2.6977219581604004
Test accuracy: 0.4586826264858246
```

3、

epochs = 30, learning_rate=0.002

epochs = 80, learning_rate=0.001

```
#  讀入測試資料並評估模型
test_ds  =  make_dataset(test_dir)
test_ds  =  test_ds.batch(batch_size)
score  =  model_best.evaluate(test_ds)
print("Test  loss:",  score[0])
print("Test  accuracy:",  score[1])
```

```
27/27 [==============================] - 2s 11ms/step - loss: 3.0661 - accuracy: 0.4790
Test loss: 3.0660834312438965
Test accuracy: 0.4790419042110443
```

4、

epochs = 30, learning_rate=0.002

epochs = 40, learning_rate=0.001

epochs = 50, learning_rate=0.0005

```
[ ]  #  讀入測試資料並評估模型
     test_ds  =  make_dataset(test_dir)
     test_ds  =  test_ds.batch(batch_size)
     score  =  model.evaluate(test_ds)
     print("Test  loss:",  score[0])
     print("Test  accuracy:",  score[1])
```

```
27/27 [==============================] - 9s 37ms/step - loss: 3.3358 - accuracy: 0.4982
Test loss: 3.3358349800109863
Test accuracy: 0.4982036054134369
```

作業心得:
一路上做了很多種嘗試，一開始前處理做了一些暴力解來解決不平衡的問題，模型跟訓練計畫的部分寫了很多種，但 Test_Acc 還是上不去.. 心很累，但不知道到底哪邊出問題，還是其實我做前處理完的資料它沒有吃到(? 還為了要算力買了 Colab pro，結果有買跟沒買的結果差不多 QQ 最後是跟同學討論之後才終於有所進展。

一開始的前處理:

```
[ ] import shutil
    from PIL import Image,ImageFilter,ImageEnhance

    # overSampling
    def overSampling():
        for i in range(0, 50):
            print(i)
            rand_bond = artists.paintings[i]-1
            while artists.paintings[i] < 150 and artists.paintings[i] < rand_bond*4 :
                rand_num = random.randint(0, rand_bond)
                rand_path = train_dir + artists.name[i] + '_' + str(rand_num) + '.jpg'
                des_path = train_dir + artists.name[i] + '_' + str(artists.paintings[i]+1) + '.jpg'
                if os.path.exists(rand_path):
                    shutil.copyfile(rand_path, des_path)
                    image = Image.open(des_path)
                    rand_trans = random.randint(0,2)
                    if rand_trans % 3 == 0:
                        image = image.transpose(Image.FLIP_LEFT_RIGHT)
                    elif rand_trans % 3 == 1:
                        image = image.transpose(Image.FLIP_TOP_BOTTOM)
                    elif rand_trans % 3 == 2:
                        image = image.transpose(Image.ROTATE_90)

                    image.save(des_path)
                    artists.paintings[i]+=1

    overSampling()
```

```
    def underSampling():
        for i in range(0, 675):
            rand = random.randint(0,876)
            target_path = train_dir + 'Vincent_van_Gogh_' + str(rand) + '.jpg'
            if os.path.exists(target_path):
                os.remove(target_path)
                print('1 success')

        for i in range(0, 500):
            rand = random.randint(0,701)
            target_path = train_dir + 'Edgar_Degas_' + str(rand) + '.jpg'
            if os.path.exists(target_path):
                os.remove(target_path)
                print('2 success')

        for i in range(0, 237):
            rand = random.randint(0,438)
            target_path = train_dir + 'Pablo_Picasso_' + str(rand) + '.jpg'
            if os.path.exists(target_path):
                os.remove(target_path)
                print('3 success')

        for i in range(0, 134):
            rand = random.randint(0,335)
            target_path = train_dir + 'Pierre-Auguste_Renoir_' + str(rand) + '.jpg'
            if os.path.exists(target_path):
                os.remove(target_path)
                print('4 success')
```

```
from sklearn.utils import class_weight

# 平衡資料
class_weight = {8: 0.2,
                13: 0.5,
                25: 4,
                29: 2,
                30: 0.25,
                34: 2,
                35: 2,
                36: 2,
                37: 2,
                38: 3,
                39: 4,
                40: 2,
                41: 3,
                49: 10}
```

一開始的模型:

```
[17] input_shape = x.shape
     # 資料增強
     data_augmentation = keras.Sequential(
         [
             layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
             layers.experimental.preprocessing.RandomRotation(0.25),
             layers.experimental.preprocessing.RandomZoom(0.2),
         ]
     )

     # 自訂你的model
     from tensorflow.keras.layers import Conv2D
     from tensorflow.keras.layers import Flatten
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.layers import Dropout
     from tensorflow.keras.layers import MaxPooling2D
     from tensorflow.keras.layers import BatchNormalization
     from tensorflow.keras.layers import GlobalAveragePooling2D

     model = keras.Sequential()

     model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', input_shape=(IMG_WIDTH, IMG_HEIGHT, 3), padding="same"))
     model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
     model.add(BatchNormalization())

     model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
     model.add(MaxPooling2D(pool_size=(2,2)))
     model.add(BatchNormalization())
```

```
model.add(BatchNormalization())

model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())

model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())


model.add(GlobalAveragePooling2D())
model.add(Dropout(rate=0.3))
model.add(Dense(256,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(rate=0.2))
model.add(Dense(num_classes, activation="softmax"))

model.summary()
```

```
input_shape = x.shape

# 自訂你的model
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import MaxPooling2D

model = keras.Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(IMG_WIDTH, IMG_HEIGHT, 3), padding="same"))
model.add(Dropout(rate=0.3))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding="same"))
model.add(Dropout(rate=0.3))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))

model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(64,activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(50, activation="softmax"))

model.summary()
```

一開始的訓練計畫:

5. 制定訓練計畫

把前處理完的資料輸入進去

```
[18] # todo
epochs = 50

# model.compile 決定learning strategy、Loss caculator
# optimizer=keras.optimizers.Adam(learning_rate=0.0005)
from keras import optimizers

adam = tf.keras.optimizers.Adam(learning_rate=0.005)

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

history = model.fit(train_ds,batch_size=batch_size,epochs=epochs,validation_data=val_ds,callbacks=[
        keras.callbacks.ModelCheckpoint("art_model_cool_best/model_{epoch:02d}",save_best_only=True)
])
```

```
Epoch 22/50
47/47 [==============================] - 37s 790ms/step - loss: 0.6125 - accuracy: 0.8045 - val_loss: 4.2743 - val_accuracy: 0.3172
Epoch 23/50
47/47 [==============================] - 37s 789ms/step - loss: 0.4995 - accuracy: 0.8457 - val_loss: 5.0318 - val_accuracy: 0.2906
Epoch 24/50
47/47 [==============================] - 37s 793ms/step - loss: 0.4428 - accuracy: 0.8615 - val_loss: 4.4571 - val_accuracy: 0.3431
Epoch 25/50
47/47 [==============================] - 37s 782ms/step - loss: 0.3437 - accuracy: 0.8918 - val_loss: 4.1102 - val_accuracy: 0.3989
```

```
[ ] # todo
epochs = 100

adam = tf.keras.optimizers.Adam(learning_rate=0.003)

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

history = model.fit(train_ds,batch_size=batch_size,epochs=epochs,initial_epoch=50,validation_data=val_ds,callbacks=[
        keras.callbacks.ModelCheckpoint("art_model_cool_best/model_{epoch:02d}",save_best_only=True)
])
```

```
Epoch 72/100
47/47 [==============================] - 38s 808ms/step - loss: 0.0394 - accuracy: 0.9889 - val_loss: 5.0382 - val_accuracy: 0.4315
Epoch 73/100
47/47 [==============================] - 38s 805ms/step - loss: 0.0297 - accuracy: 0.9907 - val_loss: 5.0293 - val_accuracy: 0.4315
Epoch 74/100
47/47 [==============================] - 38s 814ms/step - loss: 0.0263 - accuracy: 0.9912 - val_loss: 5.1700 - val_accuracy: 0.4435
Epoch 75/100
```

```
[ ] model_best=tf.keras.models.load_model('art_model_cool_best/model_54')
```

```
[ ] # todo
epochs = 150

adam = tf.keras.optimizers.Adam(learning_rate=0.00005)

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

history = model.fit(train_ds, batch_size=batch_size, epochs=epochs, initial_epoch=100, validation_data=val_ds, callbacks=[
        keras.callbacks.ModelCheckpoint("art_model_cool_best/model_{epoch:02d}", save_best_only=True)
])
```

```
Epoch 122/150
47/47 [==============================] - 39s 837ms/step - loss: 0.0373 - accuracy: 0.9882 - val_loss: 7.7560 - val_accuracy: 0.3305
Epoch 123/150
47/47 [==============================] - 40s 848ms/step - loss: 0.0413 - accuracy: 0.9870 - val_loss: 5.9696 - val_accuracy: 0.3996
Epoch 124/150
47/47 [==============================] - 38s 823ms/step - loss: 0.0351 - accuracy: 0.9885 - val_loss: 6.2971 - val_accuracy: 0.3477
Epoch 125/150
47/47 [==============================] - 39s 826ms/step - loss: 0.0359 - accuracy: 0.9882 - val_loss: 8.3452 - val_accuracy: 0.3065
```

一開始 Test_Acc 一直卡在 30 幾%

```
[ ] # 讀入測試資料並評估模型
test_ds = make_dataset(test_dir)
test_ds = test_ds.batch(batch_size)
score = model.evaluate(test_ds)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
7/7 [==============================] - 3s 366ms/step - loss: 6.6655 - accuracy: 0.3713
Test loss: 6.665514945983887
Test accuracy: 0.371257483959198
```