

資管三A

107403504

陳婕儀

# HW1-ARTCNN

連結：[https://colab.research.google.com/drive/1BuB-QW6gfRP2szpqDom\\_g7dkcTLYh7oh?usp=sharing](https://colab.research.google.com/drive/1BuB-QW6gfRP2szpqDom_g7dkcTLYh7oh?usp=sharing)

心得：資料前處理真的花的有的久的時間，加上對python的不太熟，花了更久的時間，不斷查用法，很多也會用錯，丟錯參數之類的，花了比較多時間在這。但訓練模型真的很花時間，不斷調整參數，只是怎麼調我都不知道怎樣算是好，甚至出來的test\_acc跟val\_acc之間差的真的很多，我也不太知道到底是怎麼回事，跑得有點矇，後來發現是自己路徑合併的地方參數丟錯惹，找的超久的，感覺是自己很白痴的問題。雖然我不太有能力不照to do 下去寫，所以全部都是照者to do 一步步下去做的，寫的過程中充滿著疑惑，最後有順利完成，整體心路歷程就是不斷懷疑自己，充滿問號，不過還是在帶著問號的情況下，順利完成了！

colab.research.google.com

HW1-ARTCNN-107403504.ipynb - Colaboratory

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 儲存中...

+ 程式碼 + 文字

RAM 磁碟

編輯

自己把一張現實中的圖片丟入Model做預測

```
def predictAuthor(img):
    # 寫個單圖片模型預測function
    # input : opencv img (height,width,3)
    # output : 某個作家名字 E.g. Claude_Monet
    #
    # 參考步驟:
    # 1. expand img dimension (height,width,3) -> (1,height,width,3)
    # 2. 丟入模型 model.predict
    # 3. 取出softmax後(50,) 取最大值的index作為辨識結果
    # 4. 將辨識結果轉為畫作家名字
    img = np.expand_dims(img, 0)
    predition= np.argmax(model.predict(img))
    authorName = rev_class_name [predition]
    return authorName

[ ] plt.figure(figsize=(16, 16))
for index,imgName in enumerate(show_imgs):
    imgpath = train_dir+imgName
    img = cv.imread(imgpath)
    img = cv.cvtColor(img,cv.COLOR_BGR2RGB)
    plt.subplot(4,5,index+1)
    plt.axis("off")
    plt.imshow(img)
    img = cv.resize(img, (IMG_WIDTH, IMG_HEIGHT))
    img = img / 255.0
    plt.title("True Author : {} \nPred Author : {}".format("_".join(imgName.split("_")[:-1]),predictAuthor(img)),size=11)
```

True Author : Rembrandt  
Pred Author : Rembrandt

True Author : Caravaggio  
Pred Author : Caravaggio

True Author : Pablo\_Picasso  
Pred Author : Pablo\_Picasso

True Author : Edgar\_Degas  
Pred Author : Edgar\_Degas

True Author : Leonardo\_da\_Vinci  
Pred Author : Leonardo\_da\_Vinci

執行中 (已持續 19 分鐘 43 秒) ... > ... > on\_train\_ba... > \_call\_batch... > \_call\_batch\_e... > \_call\_batch\_ho... > on\_train\_ba... > \_batch\_update... > to\_numpy\_or\_py... > map\_str... > <list... > \_to\_single\_numpy\_or... > nu... > \_nu... X



HW1-ARTCNN-107403504.ipynb ☆

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更

留言

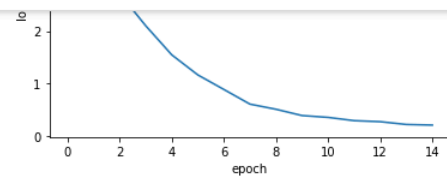
共用



+ 程式碼 + 文字

✓ RAM  
磁碟

編輯



#### ▶ # 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 [=====] - 5s 140ms/step - loss: 4.7220 - accuracy: 0.2862
Test loss: 4.721996307373047
Test accuracy: 0.2862275540828705
```


## ▼ 7. 做預測 (10分)

- 把前面讀取的圖片拿來丟入模型做預測
- 自己把一張現實中的圖片丟入Model做預測

```
[57] def predictAuthor(img):
    # 寫個單圖片模型預測function
    # input : opencv img (height,width,3)
    # output : 某個作家名字 E.g. Claude_Monet
    #
    # 參考步驟:
    # 1. expand img dimension (height,width,3) -> (1,height,width,3)
    # 2. 丟入模型 model.predict
    # 3. 取出softmax後(50,) 取最大值的index作為辨識結果
    # 4. 將辨識結果轉為畫作家名字
    img = np.expand_dims(img, 0)
    predition= np.argmax(model.predict(img))
    authorName = rev_class_name [predition]
```

✓ 10 秒 完成時間：下午2:24





HW1-ARTCNN-107403504.ipynb ☆

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更

留言 共用 設定 用戶頭像

RAM 磁碟 編輯

+ 程式碼 + 文字

# todo  
epochs = 15  
  
# model.compile 決定learning strategy、Loss caculator  
  
model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy"])  
  
history = model.fit(train\_ds,epochs=epochs,validation\_data=val\_ds)

Epoch 1/15  
188/188 [=====] - 91s 470ms/step - loss: 4.5513 - accuracy: 0.0837 - val\_loss: 3.2044 - val\_accuracy: 0.1935  
Epoch 2/15  
188/188 [=====] - 90s 473ms/step - loss: 3.2416 - accuracy: 0.1694 - val\_loss: 3.0808 - val\_accuracy: 0.2314  
Epoch 3/15  
188/188 [=====] - 90s 472ms/step - loss: 2.8517 - accuracy: 0.2633 - val\_loss: 2.9092 - val\_accuracy: 0.2586  
Epoch 4/15  
188/188 [=====] - 91s 478ms/step - loss: 2.1883 - accuracy: 0.4112 - val\_loss: 3.2018 - val\_accuracy: 0.2680  
Epoch 5/15  
188/188 [=====] - 91s 478ms/step - loss: 1.5861 - accuracy: 0.5670 - val\_loss: 3.5748 - val\_accuracy: 0.2633  
Epoch 6/15  
188/188 [=====] - 91s 477ms/step - loss: 1.2074 - accuracy: 0.6747 - val\_loss: 4.0425 - val\_accuracy: 0.2580  
Epoch 7/15  
188/188 [=====] - 91s 478ms/step - loss: 0.8911 - accuracy: 0.7618 - val\_loss: 4.1242 - val\_accuracy: 0.2773  
Epoch 8/15  
188/188 [=====] - 91s 478ms/step - loss: 0.6155 - accuracy: 0.8339 - val\_loss: 3.8940 - val\_accuracy: 0.2819  
Epoch 9/15  
188/188 [=====] - 91s 478ms/step - loss: 0.5087 - accuracy: 0.8723 - val\_loss: 4.1986 - val\_accuracy: 0.2859  
Epoch 10/15  
188/188 [=====] - 91s 478ms/step - loss: 0.3845 - accuracy: 0.9003 - val\_loss: 4.5497 - val\_accuracy: 0.3118  
Epoch 11/15  
188/188 [=====] - 92s 481ms/step - loss: 0.3473 - accuracy: 0.9072 - val\_loss: 4.4981 - val\_accuracy: 0.2906  
Epoch 12/15  
188/188 [=====] - 91s 479ms/step - loss: 0.3071 - accuracy: 0.9239 - val\_loss: 4.5774 - val\_accuracy: 0.2959  
Epoch 13/15  
188/188 [=====] - 91s 480ms/step - loss: 0.2818 - accuracy: 0.9392 - val\_loss: 4.7342 - val\_accuracy: 0.2945  
Epoch 14/15  
188/188 [=====] - 91s 479ms/step - loss: 0.2308 - accuracy: 0.9393 - val\_loss: 4.2905 - val\_accuracy: 0.2886  
Epoch 15/15  
188/188 [=====] - 91s 478ms/step - loss: 0.2032 - accuracy: 0.9501 - val\_loss: 4.6359 - val\_accuracy: 0.2832

✓ 10 秒 完成時間：下午2:24

✕

HW1 -ARTCNN

4