

國立清華大學  
智慧化企業整合

Project 3

人像年齡預測

第八組

110034558 廖文辰

授課教授:邱銘傳 博士

111/1/7

# 目錄

第一章、背景.....	1
1.1 前言.....	1
1.2 5W1H.....	1
第二章、文獻回顧.....	3
2.1 卷積神經網路(Convolutional Neural Network, CNN).....	3
第三章、研究架構及方法.....	4
3.1 資料來源.....	4
3.2 資料前處理.....	4
3.3 模型建立.....	5
3.4 載入圖片.....	7
第四章、模型訓練與訓練結果.....	11
4.1 模型訓練.....	11
4.2 訓練結果.....	12
4.3 參數優化.....	12
第五章、結論與未來展望.....	13
第六章、參考文獻.....	14

## 圖目錄

圖 1、資料前處理 .....	4
圖 2、圖片匯入 .....	4
圖 3、模型建立 .....	5
圖 4、顯示路徑與年齡 .....	6
圖 5、增加訓練速度 .....	6
圖 6、載入圖片 .....	7
圖 7、隨機抽取路徑與年齡 .....	8
圖 8、得出訓練圖像、驗證圖像及測試圖像 .....	10
圖 9、模型訓練 .....	11
圖 10、訓練結果 .....	12

# 預測人臉年齡

清華大學工業工程與工程管理學系 廖文辰

## 摘要

世界各地有形形色色的人種，每個人也有著自己獨特的樣貌，人隨著年齡增長外貌也會有著些許變化。為了增進基於年齡訊息的各種人機互動系統將在現實生活中的應用需求，本研究將建立基於人工智慧的人臉年齡預測系統，藉由卷積神經網路識別人臉圖片，並藉由優化參數降低均方根誤差。

關鍵字：圖片辨識、人像年齡預測、卷積神經網路

## 第一章、背景

### 1.1 前言

世界各地有形形色色的人種，每個人也有著自己獨特的樣貌，人隨著年齡增長外貌也會有著些許變化。在這個網路無遠弗屆的時代，人們往往會經營自己的社群網站，甚至是遊玩交友軟體，因此人們都會放上自己的大頭貼，讓別人間接認識自己。如果能更準確地估計人臉圖像的年齡，那在日常生活中基於年齡訊息的各種人機互動系統將在現實生活中有著極大的應用需求。(如預測未來自己的長相、該年齡長相與大部分人有差距，具該長相者是否存在其他健康狀況…等議題。)。本研究將以 CNN model 建立人臉年齡識別系統，幫助使用者藉由人臉照片辨識人臉年齡。除了娛樂一般民眾，對於未來此相關發展也有著重大意義。本研究使用 Kaggle 上取得之 20 至 50 歲人像資料庫進行卷積神經網路(CNN, Convolutional Neural Networks)訓練，並藉此建立人臉年齡預測系統。

### 1.2 5W1H

為預測人臉圖像的年齡，本研究使用 CNN 深度學習模型。

本研究先以 5W1H 定義問題。

Who：一般民眾、學術單位

What：更精準預估人臉年齡

Why：提高現實中人機互動系統的應用需求

When：看到照片時

Where：世界各地

How：以 CNN 深度學習建立識別模型

## 第二章、文獻回顧

### 2.1 卷積神經網路(Convolutional Neural Network, CNN)

卷積神經網路 Convolutional Neural Networks (CNN) 是一深度學習的演算法，可用於電腦視覺自動辨識、影像分類等應用上，是一以標籤資料進行模型訓練與驗證的監督式學習法。CNN 卷積神經網路模型，訓練完成後可以藉由擷取與影像形狀相關的特徵進行影像辨識 (Ketkar & Santana, 2017)。CNN 模型會從數據中自行學習特徵的識別，有別於過往的傳統影像識別方法，需要密集的進行特徵工程。CNN 的基本結構會包含多個交替堆疊的卷積層以及池化層，卷積層會使用多個濾鏡(kernels) 幫助模型擷取圖像特徵，這種構造使 CNN 架構在參數的使用上更有效率，也讓 CNN 架構的神經網路有高準確度的表現。而池化層可以降低特徵圖的維度讓網路的計算複雜度降低，可以更快的得到運算結果。

## 第三章、研究架構及方法

### 3.1 資料來源

本資料集使用 Kaggle 之 Data Every Day Dataset，資料集被劃分為訓練集與測試集。資料集中包含了 20 至 50 歲各年齡人像圖片。訓練集中有 33462 張訓練用圖片，測試集則有 7008 張影像。

### 3.2 資料前處理

插入 numpy, pandas... 等常見第三方套件與模組，並且從雲端匯入訓練集中的 33462 張訓練用圖片及測試集中的 7008 張影像。

```
import numpy as np
import pandas as pd
from pathlib import Path
import os.path

from sklearn.model_selection import train_test_split #用sklearn將train和test資料分開

import tensorflow as tf #用tensorflow建立圖片資料產生器和模型

from sklearn.metrics import r2_score #用R平方分數衡量模型的好壞
```

圖 1、資料前處理

```
from google.colab import drive
drive.mount('/content/drive')

image_dir = Path('/content/drive/MyDrive/20-50/')
from google.colab import files
```

圖 2、圖片匯入

### 3.3 模型建立

首先建立「Filepath」，以 pandas 系列的方式將資料集的图片列成清單並以字串方式呈現，這樣可透過整個子樹狀目錄查找資料集中任意 jpg 檔。

第二步建立目錄中「Age」這個路徑，具體方法是分開文件名和文件路徑的其他部分，之後再刪去文件名。lambda 是種將運算式重複運用的方式，為 pandas 系列中的每個 X 應用一個 lambda 函數，再將年齡轉成一個整數方式呈現。

第三步，將「Filepath」與「Age」合併至「image」，用樣本分數 1.0 混洗這些數據，用 random\_state 可重複展現相同的隨機結果，再用 reset\_index 讓 index(索引)重置成原本的樣子。

```
filepaths = pd.Series(list(image_dir.glob(r'*/*.jpg')), name='Filepath').astype(str)
ages = pd.Series(filepaths.apply(lambda x: os.path.split(os.path.split(x)[0])[1], name='Age').astype(np.int)
images = pd.concat([filepaths, ages], axis=1).sample(frac=1.0, random_state=1).reset_index(drop=True)
```

圖 3、模型建立

	Filepath	Age
0	/content/drive/MyDrive/20-50/train/22/113386.jpg	22
1	/content/drive/MyDrive/20-50/train/29/170287.jpg	29
2	/content/drive/MyDrive/20-50/train/41/179421.jpg	41
3	/content/drive/MyDrive/20-50/train/47/126188.jpg	47
4	/content/drive/MyDrive/20-50/train/30/167003.jpg	30
...	...	...
40435	/content/drive/MyDrive/20-50/train/27/174935.jpg	27
40436	/content/drive/MyDrive/20-50/train/49/144643.jpg	49
40437	/content/drive/MyDrive/20-50/train/25/147638.jpg	25
40438	/content/drive/MyDrive/20-50/train/33/146050.jpg	33
40439	/content/drive/MyDrive/20-50/train/49/142203.jpg	49

40440 rows × 2 columns

圖 4、顯示路徑與年齡

讓 image 只留 5000 筆，以增加訓練速度，再將 train 與 test 以 7:3 的比例分割。

且建立訓練集和測試集，一樣隨機抽取數據並用 random\_state 可重複展現相同的隨機結果。

```
# 只使用5000張圖片是因為縮短訓練時間
image_df = images.sample(5000, random_state=1).reset_index(drop=True)

train_df, test_df = train_test_split(image_df, train_size=0.7, shuffle=True, random_state=1)
```

圖 5、增加訓練速度

### 3.4 載入圖片

用 Rescale 將圖片的值縮放到 0~1。

validation\_split 用於在沒有提供驗證集的時候，按 0.2 的比例從訓練集中取出一部分作為驗證集。

```
train_generator = tf.keras.preprocessing.image.ImageDataGenerator(  
    rescale=1./255,  
    validation_split=0.2  
)  
  
test_generator = tf.keras.preprocessing.image.ImageDataGenerator(  
    rescale=1./255  
)
```

圖 6、載入圖片

train\_df

	Filepath	Age
1334	/content/drive/MyDrive/20-50/train/26/162593.jpg	26
4768	/content/drive/MyDrive/20-50/train/28/147733.jpg	28
65	/content/drive/MyDrive/20-50/test/45/42885.jpg	45
177	/content/drive/MyDrive/20-50/train/27/178438.jpg	27
4489	/content/drive/MyDrive/20-50/test/31/44435.jpg	31
...	...	...
2895	/content/drive/MyDrive/20-50/train/25/166220.jpg	25
2763	/content/drive/MyDrive/20-50/train/23/174142.jpg	23
905	/content/drive/MyDrive/20-50/test/48/43433.jpg	48
3980	/content/drive/MyDrive/20-50/test/33/42502.jpg	33
235	/content/drive/MyDrive/20-50/train/22/114475.jpg	22

3500 rows × 2 columns

圖 7、隨機抽取路徑與年齡

為了再加快訓練時間，將圖像的分辨率降低到 120\*120。因為圖片是彩色，color\_mode 設為 RGB。為了不變動 generator，class\_mode 設為 raw，本研究將使用與「age」進行比較。先在此指定 batch\_size，之後就不必在模型中調整。為了隨機抽取訓練數據，shuffle=True。seed 的數字除了 0 或是未填寫以外，都是為了確保都是以相同方式抽取數據。最後將子集設為 training。

validation 步驟大致上與 train 相同，只差在子集不同。

Test 的部分，因為要評估模型的成果，不需要子集也不用隨機抽取，故刪除 shuffle 和 subset。

最後得出有 2800 張訓練圖像、700 張驗證圖像及 1500 張測試圖像。

```

train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Age',
    target_size=(120, 120),      #像素縮到120*120 加快訓練時間
    color_mode='rgb',
    class_mode='raw',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='training'
)

val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Age',
    target_size=(120, 120),
    color_mode='rgb',
    class_mode='raw',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='validation'
)

test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='Filepath',
    y_col='Age',
    target_size=(120, 120),
    color_mode='rgb',
    class_mode='raw',
    batch_size=32,
    shuffle=False                #不用重新抽取訓練，因為已經要實際去測試資料
)

```

```

Found 2800 validated image filenames.
Found 700 validated image filenames.
Found 1500 validated image filenames.

```

圖 8、得出訓練圖像、驗證圖像及測試圖像

## 第四章、模型訓練與訓練結果

### 4.1 模型訓練

前置步驟準備完，第一層卷積層的 filter 為 16，kernel size 為 3\*3，第二層卷積層 filter 為 32，kernel size 為 3\*3，所有的 activation 皆設為 relu。將圖中參數設定完後即可進行模型訓練。

```
inputs = tf.keras.Input(shape=(120, 120, 3))
x = tf.keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu')(inputs)
x = tf.keras.layers.MaxPool2D()(x)
x = tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPool2D()(x)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(64, activation='relu')(x)
x = tf.keras.layers.Dense(64, activation='relu')(x)
outputs = tf.keras.layers.Dense(1, activation='linear')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)

model.compile(
    optimizer='adam',
    loss='mse'
)

history = model.fit(
    train_images,
    validation_data=val_images,
    epochs=100,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=5,
            restore_best_weights=True
        )
    ]
)
```

圖 9、模型訓練

## 4.2 訓練結果

本研究使用 RMSE 和判定係數評估整個模型的好壞。

```
predicted_ages = np.squeeze(model.predict(test_images))
true_ages = test_images.labels

rmse = np.sqrt(model.evaluate(test_images, verbose=0))
print("      Test RMSE: {:.5f}".format(rmse))

r2 = r2_score(true_ages, predicted_ages)
print("Test R^2 Score: {:.5f}".format(r2))

      Test RMSE: 8.82989
Test R^2 Score: -0.00056
```

圖 10、訓練結果

如上圖可知，RMSE 為 8.82989，代表模型中的每個預測值與真實年齡平均差了約 8.8 歲，判定係數為-0.00056，因判定係數的範圍落在-1 與 1 之間，以年齡預測來說算不太準確。

## 4.3 參數優化

為了降低均方根誤差，本研究嘗試調整 epochs，結果如下圖所示。

epoch	100	50	10
RMSE	8.82989	8.82743	8.84744

## 第五章、結論與未來展望

本研究利用 CNN 的方法將 20-50 歲的不同人臉進行分類，最後預測結果與真實年齡平均差了 8.8 歲，即使經過優化，RSME 也只有些微下降，可能原因有卷積層只有 2 層、使用的圖像資料不夠多... 等，網路上也已經有許多年齡預測 app，各種 app 預測出來的準確度也不盡相同，因為各家資料庫中的照片也不盡相同。年齡預測的演算法也會依光線、拍攝角度、妝容、種族、照片顏色等因素差異而有所不同，若臉部識別的技術更加成熟，未來也許能對性取向、犯罪傾向與暴力傾向等議題之預測能有更進一步的發展。

## 第六章、參考文獻

Ketkar, N., & Santana, E. (2017). Deep Learning with Python (Vol. 1): Springer. 15