

國立臺灣師範大學附屬高級中學第46屆科學展覽會

作品說明書封面

科 別：

組 別：

作品名稱：大腸鏡影像檢測有無息肉

關 鍵 詞：人工智慧、VGG-16、息肉（最多3個）

編 號：

摘要

由於尚未得到教授欲提供的腸道腫瘤照片資料，因此先暫時以 cifar10進行訓練並建立其 VGG-16的模型，又因其中的前幾層都相同，因此後續待拿到資料後即可以進行轉移學習。

壹、研究動機

由於醫生經常要耗費大量時間及精力來人工辨識病人腸道胃鏡之類的照片，辨識是否有息肉的產生，為了盡我們所能減少醫生的辛勞，因此我們希望利用 CNN 來進行機器學習並直接辨識該大腸鏡照片中是否有息肉，以減少人工辨識所需浪費的人力及時間。谷歌公司曾利用 C2D2深度學習演算法為大腸3D 建模提高檢查的覆蓋面積。配合我們研究的大腸鏡息肉檢測模型正是如虎添翼。

貳、研究目的

藉由深度學習來協助進行大腸鏡影像有無息肉的辨識。

參、研究設備及器材

（一）、訓練資料

1. cifar10

以預先訓練模型中的前幾層，利用轉移學習提高訓練效率。

2. 大腸鏡息肉影像

以訓練模型主體。

(二)、筆電

1. 2015 Mac Book PRO

2. 2020 Mac Book PRO

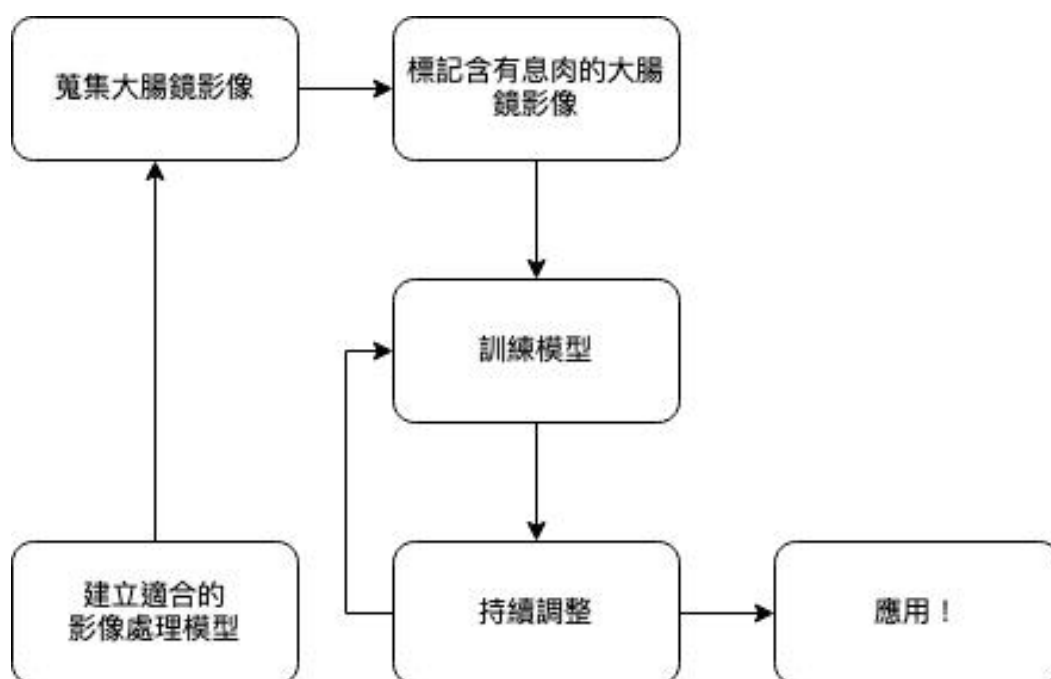
本研究使用的研究器材。

(三)、開發環境

1. Google Colab 免費方案

本研究使用的開發環境，免費又好用，可以線上共用。其提供12Gb 的隨機存取記憶體，與100Gb 的硬碟儲存訓練資料。但是每次使用時必須自行引入所有檔案，所有程式碼區塊共用命名空間。

肆、研究過程或方法



以上為我們研究的流程圖。

一、建立適合的影像處理模型

在目前這個階段我們尚未取得大腸鏡息肉照片供我們訓練。但是我們可以利用我們將使用的 VGG-16模型的特性「轉移學習」先行訓練模型的前幾層。

我們使用 keras 建立並且訓練模型，我們使用的模型是以 VGG-16為基礎，對我們的需求更進一步修改的模型。前面有提到因為要使用 cifar10的資料做轉移學習，故取得轉移學習資料時的模型共有10個輸出的神經元。此模型圖可以在附錄（一）找到。

（一）、VGG-16

至於使用 VGG-16的原因，不外乎是因為他擁有優異的影像分類能力，在原先的設計中能夠分類1000種不同的物品。所以可以在他的模型最後面看到最後一個1000個神經元的全連接層以 softmax 作為激活函數。VGG-16之所以後面或有一個16是因為 VGG-16有13個卷積層和3個全連接層。可以參考附錄（二）的 VGG 實驗圖，當中的 D 和 E 就是 VGG-16和 VGG-19。之所以選擇 VGG-16而不選擇 VGG-19是因為我們認為我們欲訓練的模型沒有大到需要19層，畢竟他原先是設計分類1000種物品，而我們只是要辨識有無息肉。此模型甚至利用了一個名為”Multiple Scale training”的技術會在訓練時隨機切取原始圖片[論文一]，預測不同的裁切大小並且在各個位置各取一張，最後取平均成為預測結果。雖然此論文的結論是越深的模型越好，但是論文中也有提到，越深的模型就會需要越多的資料訓練，否則模型多再多層結果也相差不大。

論文中提到他在訓練時使用了 transfer learning，將第一個訓練的模型作為之後

模型前四層的初始權重。因為前幾層的卷基層辨識的都是點線之類的幾何特徵，所以可以轉移。

（二）、轉移學習

因為大腸鏡息肉的樣本有限，故我們將會使用 cifar10 的資料訓練本模型的前幾層以進行轉移學習，提高準確度。

二、蒐集大腸鏡影像

本研究與大學端合作，即將取得一些大腸鏡影像。

三、標記含有息肉的大腸鏡影像

本研究與大學端合作，即將取得一些標記好的大腸鏡影像。

四、訓練模型

（一）、轉移學習

本研究在訓練模型時遭遇了相當多的困難，特別是在第一次訓練時。我們為了要取得可以拿來轉移學習的模型，必須要先用 cifar10 的資料進行訓練。但是在訓練資料提取的程式碼和模型輸入的程式碼過程中遇到了不少的問題，最後終於成功了。

當中我們使用 `categorical_crossentropy` 作為我們模型的損失函數，該損失函數適合分類的問題，而其特性為驗證答案的輸入端須為所有可能數的維度的向量。

首先讀取資料後，必須將其轉為三層的向量，並且將所有像素的 RGB 轉為 0 到 1 的浮點數，然後和所有其他的圖片經過處理後的向量串在一起成為訓練集的列表。同時處

理圖片的標記，因為 cifar10共有十類圖片，將所有圖片的標記轉為一個十維的向量，如下圖，表示第一類圖片的標記。

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

詳細程式碼請看附錄（三）。

而我們替 cifar10特製的模型可以在附錄（六）找到。當中將原本 VGG-16的最後一層1000個神經元的全連接層換成10神經元的以切合需求。

五、持續調整

一開始我們使用 `harse_categorical_crossentropy` 作為學習使用的損失函數，但是後來發現此損失函數並非最理想的。因為偏差值一直很大。所以將其調整為 `categorical_crossentropy`。

我原先以為就算不修改訓練時的順序，應該對模型訓練沒有關係，因為此模型並沒有長期記憶或是短期記憶的問題。但是在開始訓練後我觀察到正確率極大幅度的上升，原來是模型以為答案只有這一種，所以那個答案的機率特別高。但是就在我把順序打亂後，模型的正確率急遽下降，逐漸逼近0.1，相當於亂猜，我們初步認為是訓練用的照片太不清楚，其實人也看不太出來那些是什麼。



例如說這張，看起來有點像吹風機，也有點像藍色小手槍，但是這張圖片其實是飛機。像是這樣的圖片品質我們很難訓練出高品質的模型。同樣的例子還有以下這個。



乍看之下像綿羊，細看像老鼠，誰也沒想到這是一隻鳥，我們需要增進我們模型的品質。

六、應用

本研究最主要的用途即為協助醫生進行辨識，由於訓練後得到的準確度並非100%，因此在辨識上仍然有極小的機率進行誤判，為了確保不會因誤判而造成延誤治療，因此還是同時還需要醫生的判斷。不過我們認為應該在醫生檢查過後，再進行辨識，以確保判斷得到的結果與醫生相同，而不要讓醫生因為已經有了電腦判斷的結果而產生主觀意見，使他無意中忽略了電腦沒判斷到的部分。我們也希望這個技術可以因此造福大

眾，減少更多因誤判導致的延誤治療。不過若真的要從根本解決息肉的問題，注意飲食及避免煙、酒才是至關重要的解決方法。

伍、研究結果

陸、討論

一、文獻探討

在我們的研究中，我們使用的 VGG 是 Oxford 的 Visual Geometry Group 組提出的。此模型證明了在某種程度上越深的模型的效果越好。在人工智慧進行大腸保健這方面上，已經有谷歌公司為病人的大腸3D 建模提高醫生人工檢查的覆蓋率，目前無人提出開源的人工智慧模型檢測有無息肉。

柒、結論

捌、參考資料及其他

一、參考資料

Karen Simonyan & Andrew Zisserman. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. International Conference on Learning Representations

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. (1998). *Gradient-based learning applied to document recognition*, in Proceedings of the IEEE

, vol. 86, no. 11, pp. 2278-2324, doi: 10.1109/5.726791

荷葉田田 (2019 年 03 月 12 日)。VGG16和 VGG19介紹【部落格影音資料】。取自

<https://blog.csdn.net/qian2213762498/article/details/88422941>

I code so I am (2017 年 12 月 19 日)。CNN 經典模型應用【部落格影音資料】。取自

<https://ithelp.ithome.com.tw/articles/10192162>

二、附錄

特別需要提及的是由於我們開發環境的關係（Google Colab），所有程式共用命名空間，其特性更方便開發者進行人工智慧相關的開發。故下列程式碼中各函數與變數名稱是共用的。

(一)、模型結構1

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
=====		
conv2d_163 (Conv2D)	(None, 28, 28, 64)	1792
conv2d_164 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d_63 (MaxPooling)	(None, 14, 14, 64)	0
conv2d_165 (Conv2D)	(None, 14, 14, 128)	73856
conv2d_166 (Conv2D)	(None, 14, 14, 128)	147584
max_pooling2d_64 (MaxPooling)	(None, 7, 7, 128)	0
conv2d_167 (Conv2D)	(None, 7, 7, 256)	295168
conv2d_168 (Conv2D)	(None, 7, 7, 256)	590080
conv2d_169 (Conv2D)	(None, 7, 7, 256)	590080
max_pooling2d_65 (MaxPooling)	(None, 3, 3, 256)	0
conv2d_170 (Conv2D)	(None, 3, 3, 512)	1180160
conv2d_171 (Conv2D)	(None, 3, 3, 512)	2359808
conv2d_172 (Conv2D)	(None, 3, 3, 512)	2359808
max_pooling2d_66 (MaxPooling)	(None, 1, 1, 512)	0
conv2d_173 (Conv2D)	(None, 1, 1, 512)	2359808
conv2d_174 (Conv2D)	(None, 1, 1, 512)	2359808
conv2d_175 (Conv2D)	(None, 1, 1, 512)	2359808
max_pooling2d_67 (MaxPooling)	(None, 1, 1, 512)	0
flatten_12 (Flatten)	(None, 512)	0
dense_36 (Dense)	(None, 512)	262656
activation_3 (Activation)	(None, 512)	0
batch_normalization_2 (Batch Normalization)	(None, 512)	2048
dense_37 (Dense)	(None, 512)	262656
activation_4 (Activation)	(None, 512)	0
batch_normalization_3 (Batch Normalization)	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_38 (Dense)	(None, 10)	5130
activation_5 (Activation)	(None, 10)	0
=====		

(二)、VGG 實驗圖

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

(三)、產生訓練資料之程式碼

產生訓練資料用

```
import os
```

```
data_path = '/content/sample_data/cifar10/train/'
```

```
Xtrain = []
```

```
Ytrain = []
```

```
classes = sorted(os.listdir(data_path))
```

```
print(f'classes: {classes}')
```

```
class_num = 0
```

```
for img_class in classes:
```

```
    for img in os.listdir(f'{data_path}{img_class}/'):
```

```
        image = load_img(f'{data_path}{img_class}/{img}',target_size=(32,32))
```

```
        image = np.array(image)
```

```
        image = image.astype('float32')
```

```
        image /= 255
```

```
        Xtrain.append(image)
```

```
        Ytrain.append([0]*class_num + [1] + [0]*(9-class_num))
```

```
    class_num += 1
```

```
Xtrain = np.array(Xtrain)
```

```
Ytrain = np.array(Ytrain)
```

(四)、預測圖片之程式碼

```
# 此格為測試模型用
```

```
test_img_path = '/content/sample_data/10020.jpg'
```

```

image = load_img(test_img_path,target_size=(32,32))
image = np.expand_dims(image, axis=0)
# image = np.array(image)
image = image.astype('float32')
image /= 255

features = model.predict(image)
print(features)
#print('Predicted:', decode_predictions(features, top=3)[0])

```

(五)、訓練模型之程式碼

```

EPOCH = 3
history = model.fit(
    Xtrain,
    Ytrain,
    epochs=EPOCH
)

```

(六)、建立模型之程式碼（包含編譯，設定損失函數等等）

```

# 此格為建立模型用，再度執行會覆寫 model

from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
import numpy as np

```

```

from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array

image = load_img('/content/sample_data/10020.jpg',target_size=(32,32))

# image = np.expand_dims(image, axis=0)

image = np.array(image)

image = image.astype('float32')

image /= 255

# print(image)

input_shape = (32, 32, 3)

model = Sequential([
    Conv2D(64, (3, 3), input_shape=input_shape, padding='same',
        activation='relu'),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(512, (3, 3), activation='relu', padding='same'),

```

```

Conv2D(512, (3, 3), activation='relu', padding='same'),
Conv2D(512, (3, 3), activation='relu', padding='same'),
MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
Conv2D(512, (3, 3), activation='relu', padding='same'),
Conv2D(512, (3, 3), activation='relu', padding='same'),
Conv2D(512, (3, 3), activation='relu', padding='same'),
MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
Flatten(),
Dense(4096, activation='relu'),
Dense(4096, activation='relu'),
Dense(10, activation='softmax')
])

```

```

model.summary()

```

```

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

```