以 CNN 辨識瘧疾細胞

國立清華大學 黃子芸

指導教授：邱銘傳

目錄

一、摘要

在資源較不足的地方沒有那麼多的專業人士能透過顯微鏡來判斷病人是否得到瘧疾，而在瘧疾已不再流行的地區，醫護人員對此疾病不熟悉，診斷時可能也會忽略這個可能性，因此本研究透過建立 CNN 模型的方式來辨識細胞是否感染瘧疾，經過參數調整後，能使辨識準確率達到 95.8%，進而達到輔助醫護人員辨識的功能。

二、介紹

1.簡介

瘧疾是一種致命的疾病，由寄生蟲引起，透過蚊子叮咬傳播給人們。瘧疾引起的症狀通常包括發燒，疲倦，嘔吐和頭痛。 在嚴重的情況下，它可能導致皮膚發黃，癲癇發作，昏迷或死亡。

2018 年，世界衛生組織報告指出[1]，全球估計有 2.19 億瘧疾病例，死亡 435,000，其中 93％發生在撒哈拉以南非洲，這是與傳染病有關的第二大死亡原因。在資源較不足的地方沒有那麼多的專業人士能透過顯微鏡來判斷病人是否得到瘧疾，而在瘧疾已不再流行的地區，醫護人員對此疾病不熟悉，診斷時可能也會忽略這個可能性，因此透過深度學習的方式來辨識細胞是否感染瘧疾，不但可以提高辨識準確率，亦能減輕醫護人員的負擔，也希望能進而降低因醫護人員判斷錯誤而死亡的病例。

許多論文中也有透過深度學習來辨識細胞的前例，像是檢測通過廣角顯微鏡記錄的幼蟲斑馬魚腦 z 堆棧圖像中含酪氨酸羥化酶（TH 標記）的細胞[2]，以及乳腺癌細胞的有絲分裂檢測[3]…等，由此得知以 deep learning 的方式來協助醫療上的圖片辨識，確實是一個可行的方法。

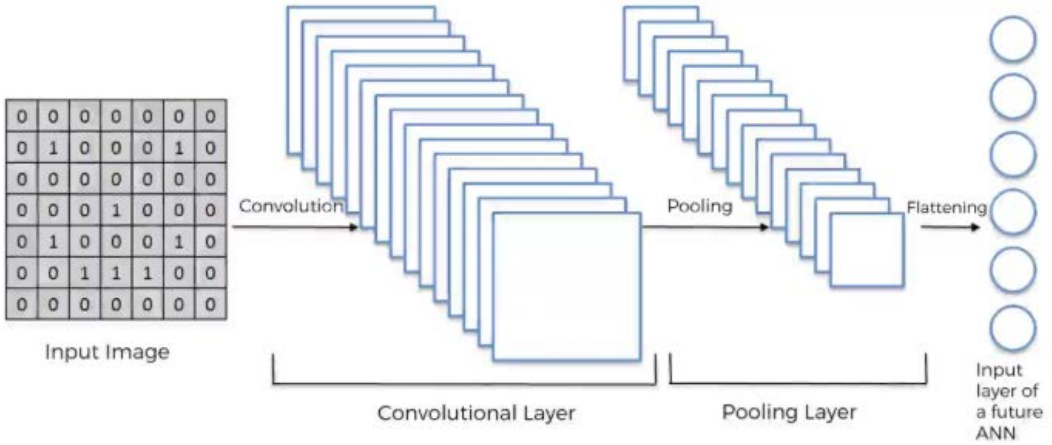
2. 5W1H

項目	內容
What	以細胞的圖片來分辨其是否感染瘧疾

Where	醫護站、醫院
Who	醫護人員
When	需要檢驗病患是否感染瘧疾時
Why	透過電腦來分辨細胞是否受瘧疾感染，能提供醫護人員辨識瘧疾上的協助，降低因判斷錯誤而死亡的病例。
How	CNN 模型訓練

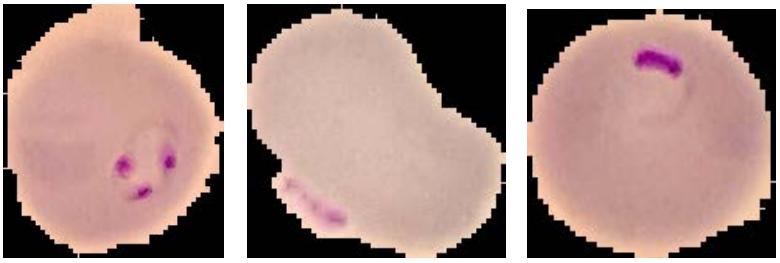
三、研究方法

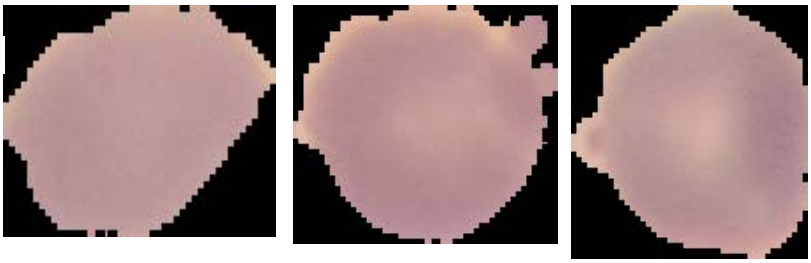
CNN（Convolutional Neural Network，卷積神經網絡）是模仿人類大腦認知方式的一種深度學習方法，例如我們辨識一個圖像，會先注意到顏色鮮明的點、線、面，之後再將它們構成一個個不同的形狀(眼睛、鼻子、嘴巴...)，這種抽象化的過程就是 CNN 演算法建立模型的方式。 CNN 透過卷積層(Convolution Layer)、池化層(Pooling Layer)、全連接層(Fully Connected Layer) 這三個主要步驟來達到辨識圖片的功能。卷積層就是由點的比對轉成局部的比對，透過一塊塊的特徵研判，逐步堆疊綜合比對結果，Max Pooling 的概念是挑出矩陣當中的最大值，使得若圖片平移幾個 Pixel 對判斷上完全不會造成影響，以及有很好的抗雜訊功能，最後則是透過全連接層將 Matrix 拉值，過程如下圖[4]。



四、個案研究

此 data set 共有 27,558 張圖片，被感染的細胞以及未被感染的細胞各半(但我的電腦跑不動，所以在 train model 時只各自讀了 3000 張圖片)。

圖片如下: Parasitized - 

Uninfected – 

1. 設定圖片大小及標註

首先將圖片大小縮小為 50*50，再將已受感染的細胞圖片 label 設為 1，未被感染的細胞圖片 label 設為 0。

- Code >

```
infected_path = base_path +'Parasitized/'
for file in listdir(infected_path):
    if a<=total_a:
        if file.endswith('.png'):
            file_path = infected_path + file
            image = imageio.imread(file_path)
            image = cv2.resize(image,(IMG_SHAPE,IMG_SHAPE)).astype('float32')/255.0
            infected_cells.append(image)
            cell_images.append(image)
            cell_labels.append(1)
            a+=1


uninfected_path = base_path +'Uninfected/'
for file in listdir(uninfected_path):
    if b<=total_b:
        if file.endswith('.png'):
            file_path = uninfected_path + file
            image = imageio.imread(file_path)
            image = cv2.resize(image,(IMG_SHAPE,IMG_SHAPE)).astype('float32')/255.0
            uninfected_cells.append(image)
            cell_images.append(image)
            cell_labels.append(0)
            b+=1
```

2. 將圖片順序打亂

將圖片的順序打亂後才會比較接近真實的概率，也避免有 overfitting 的情形發生，也能避免同一個組合的 batch 反覆出現，使的模型記住這些順序。

- Code >

```
def reorder(old_list,order):
    new_list = []
    for i in order:
        new_list.append(old_list[i])
    return new_list

np.random.seed(seed=42)
indices = np.arange(len(cell_labels))
np.random.shuffle(indices)
indices = indices.tolist()
cell_labels = reorder(cell_labels,indices)
cell_images = reorder(cell_images,indices)


#順序打亂後image、label各存入一個array

image_array = np.array(cell_images)
label_array = np.array(cell_labels)
```

3. MODEL

Step1：將所有圖片之 10%為 testing，而 validating 的數量與 testing 同，而這三個集合不能有交集，因此 training 數量為所有圖片的 80%。

■ Code >

```
X_train, X_test, y_train, y_test = train_test_split(image_array, label_array, train_size=0.90, random_state=100)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=len(y_test), random_state=100)
```

```
size of training data set 4800
size of validating data set 600
size of testing data set 600
```

Step2：建立 CNN 模型。

■ Code >

```
model =  Sequential([

    #convolutional layers
    Conv2D(32, (3,3), activation='relu', input_shape=(50,50, 3),padding='same'),
    MaxPooling2D(2, 2),
    Conv2D(64, (3,3), activation='relu',padding='same'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu',padding='same'),
    MaxPooling2D(2,2),

    # matrix拉直後加入隱藏層
    Flatten(),
    Dropout(0.50),
    Dense(128, activation='relu'),
    #二元分類，因此採用sigmoid
    Dense(1, activation='sigmoid')
])
```

```
# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=[tf.keras.metrics.binary_accuracy])
model.summary()

#進行訓練, 訓練過程會存在 history 變數中
epochs = 20
batch_size =  300
history = model.fit(X_train,y_train,
    steps_per_epoch=int(np.ceil(len(y_train)/ float(batch_size))),
    epochs=epochs,
    validation_data=(X_val,y_val),
    validation_steps=int(np.ceil(len(y_val) / float(batch_size)))
)
```

4. Graph

Step1：繪製 accuracy 及 loss 圖表，並以 ROC 分析模型優劣，ROC 曲線中，將 FPR(實際上被感染，判斷正確)定義為 X 軸，TPR(實際上未被感染，判斷成被感染)定義為 Y 軸。此方法適合用在二元分類模型，AUC 就是 ROC 曲線下的面積，當 AUC 的數值越高，代表效能越好[5]。

■ Code >

```
#做ROC分析，此方法適合用在二元分類模型，AUC就會是ROC曲線下的面積
def GetFalseTruePositiveRate(y_true,y_prob,threshold):

    y_predict = np.fromiter([1 if x > threshold else 0 for x in y_prob ],int)
    n_positives = y_true.sum()
    n_negatives = y_true.shape[0] - n_positives

    n_true_pos = 0
    n_false_pos = 0
    for pred_value,true_value in zip(y_predict,y_true):
        # true positive 實際上被感染，辨識成功
        if true_value == 1 and pred_value == 1:
            n_true_pos += 1
        # false positive 實際上未被感染，辨識成被感染
        elif true_value == 0 and pred_value == 1:
            n_false_pos += 1
    #tpr=tp/(tp+fn)
    true_pos_rate = n_true_pos/n_positives
    #fpr=fp/(fp+tn)
    false_pos_rate = n_false_pos/n_negatives
    return false_pos_rate,true_pos_rate
```

Step2：透過 ROC 分析模型建立混淆矩陣，可以更清楚的知道 CNN 模型

判斷每張圖片的結果與真實值是否一致。

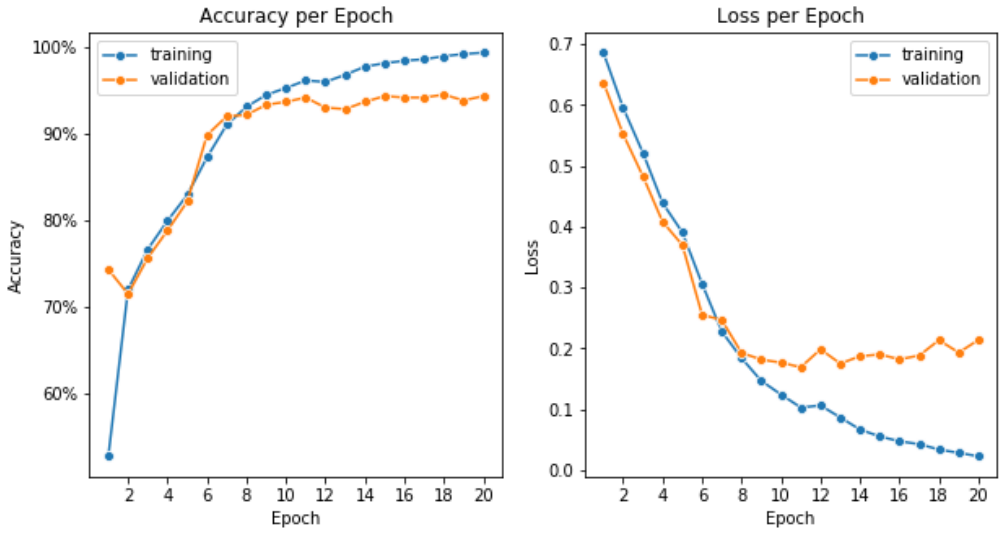
■ Code >

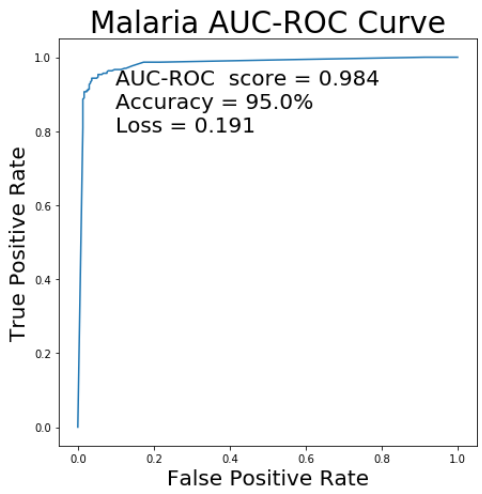
```
def MakeConfusionMatrix(y_true,y_prob,threshold):
    confusion_matrix = np.array([[0,0],[0,0]])
    for pred_value,true_value in zip(y_prob,y_true):
        if true_value == 1:
            #true positive
            if pred_value > threshold:
                confusion_matrix[0,0] += 1
            #false negative
            else:
                confusion_matrix[1,0] += 1
        else:
            #false positive
            if pred_value > threshold:
                 confusion_matrix[0,1] += 1
            #true negative
            else:
                confusion_matrix[1,1] += 1
    fig = plt.figure(figsize=(5,5))
    ax =  fig.gca()
    sns.heatmap(confusion_matrix,ax=ax,cmap='Blues',annot=True,fmt='g',
               xticklabels = ['Infected','Uninfected'],
               yticklabels=['Infected','Uninfected'])
    ax.set_ylabel('Actual',fontsize=20)
    ax.set_xlabel('Predicted',fontsize=20)
    plt.title('Confusion Matrix',fontsize=24)
    plt.show()
```

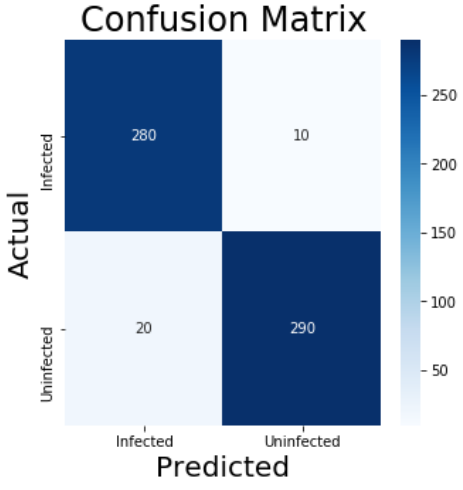
五、 研究結果

1. OUTPUT

CNN 模型(3 層 Convolution Layer、3 層 Pooling Layer、兩層 Dense Layer)之準確率以及 Loss 可以由下方圖表看出,學習越多回合其準確率就會隨之提升,而 Loss 也能逐漸下降，透過 ROC 分析，也可以發現此模型的效能非常高，準確率也高達 95%,而從混淆矩陣中更可以清楚得知此模型在 600 張驗證圖片中，有 570 張是準確辨識的，有 10 張是已感染細胞辨識成未被感染，20 張是未被感染的細胞辨識成已被感染。







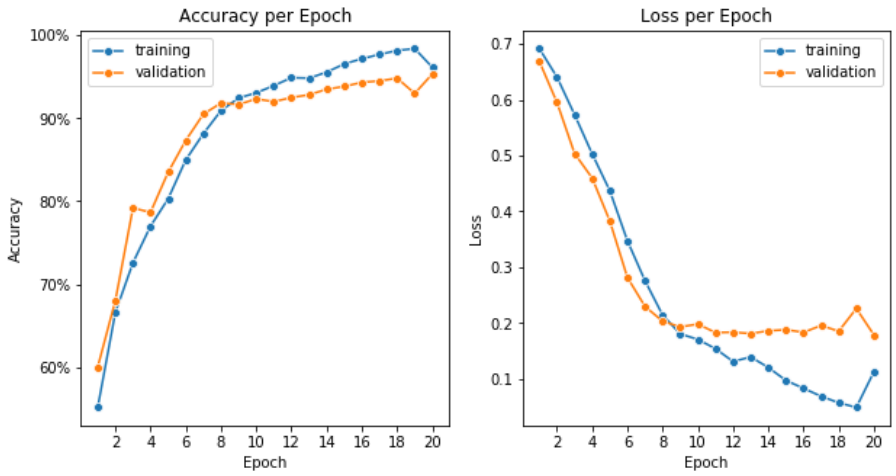
2.參數調整

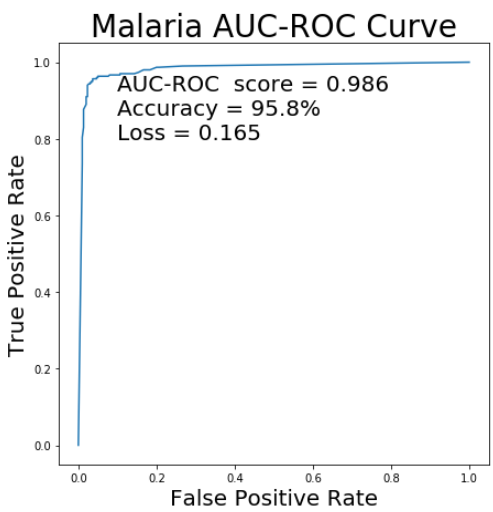
由於電腦性能不佳，無法以 train 更多圖片的方式來增加準確率，因此希望能透過調整參數的方式來提升準確率。在 epoch 皆為 20 的情況下，試著以添加 Convolution Layer、Pooling Layer、Dense Layer 以及更改 batch 的方式，來增加準確率，並與原始模型進行比較。

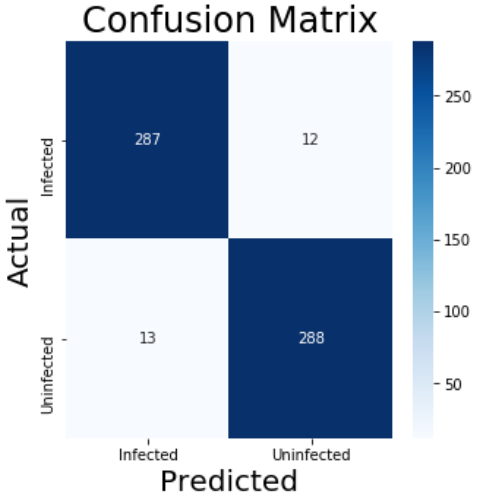
	Convolution Layer	Pooling Layer	Dense Layer	batch	AUC-ROC Score	Loss	準確率
original	3 層	3 層	2 層	300	0.984	0.191	95%
Test1	3 層	3 層	3 層	300	0.977	0.263	94.2%
Test2	4 層	4 層	2 層	300	0.986	0.165	95.8%
Test3	4 層	4 層	3 層	300	0.984	0.21	94.5%
Test4	4 層	4 層	2 層	200	0.985	0.162	95.7%
Test5	4 層	4 層	2 層	400	0.984	0.156	94%
Test6	5 層	5 層	2 層	300	0.987	0.16	95%

從表格中可以看出，更動參數過後，其實準確率差異不大，而當 Dense 增加時，準確率皆些微下降，batch 增加或減少都使得準確率稍微下降，因此決定維持 300 不變，做完所有測試後發現，在 Convolution Layer=3 層、Pooling Layer=3 層、Dense Layer=2 層、batch=300 時，準確率是所有測試中最高的。

- Test2







3.結論及未來發展

CNN 模型的辨識準確率在所有測試中均能達到 90%以上，雖然已經是相當高了，但如果真的要實際應用的話，可能還是要追求到至少 99%以上，或許可以嘗試再 trian 更多圖片，或是再繼續調整其他參數來達到目標!

REFERENCES

[1] World Health Organization, World Malaria Report 2018, C. D. Jones, A. B. Smith, and E.F. Roberts, *Book Title*, Publisher, Location, Date.

[2] DEEP LEARNING FOR AUTOMATIC CELL DETECTION IN WIDE-FIELD MICROSCOPY ZEBRAFISH IMAGES, Bo Dong, Ling Shao, Marc Da Costa,Oliver Bandmann ,Alejandro F Frangi

[3] Beyond Classification: Structured Regression for Robust Cell Detection Using Convolutional Neural Network, Yuanpu Xie, Fuyong Xing, Xiangfei Kong, Hai Su1, and Lin Yang

[4] https://dotblogs.com.tw/greengem/2017/12/17/094150

[5] https://ithelp.ithome.com.tw/articles/10229049?sc=rss.qu