癌疾細胞辨識

108034537 黃子芸

Contents

5W1H

What

以細胞的圖片來分辨其是否感染瘧疾

When

需要檢驗病患是否感染瘧疾時

Where

醫護站、醫院

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。

Who

醫護人員

How

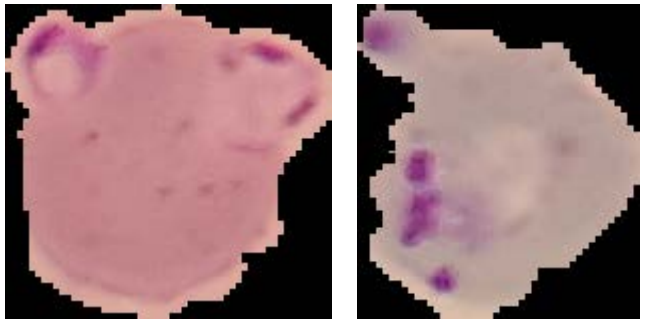
CNN模型訓練

Dataset

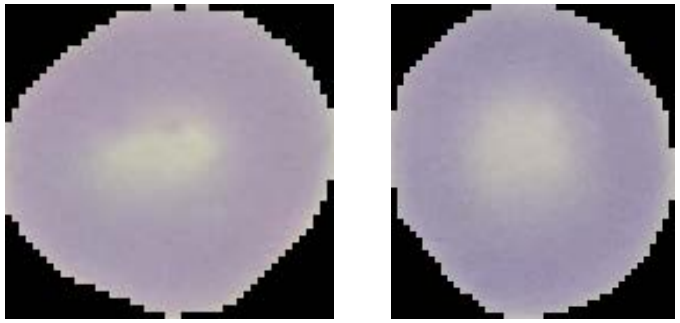
- Dataset資料分為兩個資料夾

名稱	修改日期	類型	大小
Parasitized	2019/12/30 下午 ...	檔案資料夾	
Uninfected	2019/12/31 下午 ...	檔案資料夾	

- Infected



- Uninfected



- Infected及Uninfected分別讀取3000張

Dataset

- 讀入資料夾
- 設定圖片大小、標註label

>Code

```
IMG_SHAPE = 50

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
```

Dataset

- 將圖片順序打亂

>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)
```

- Training : Testing : validation= 8 : 1 : 1

```
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
```

Model

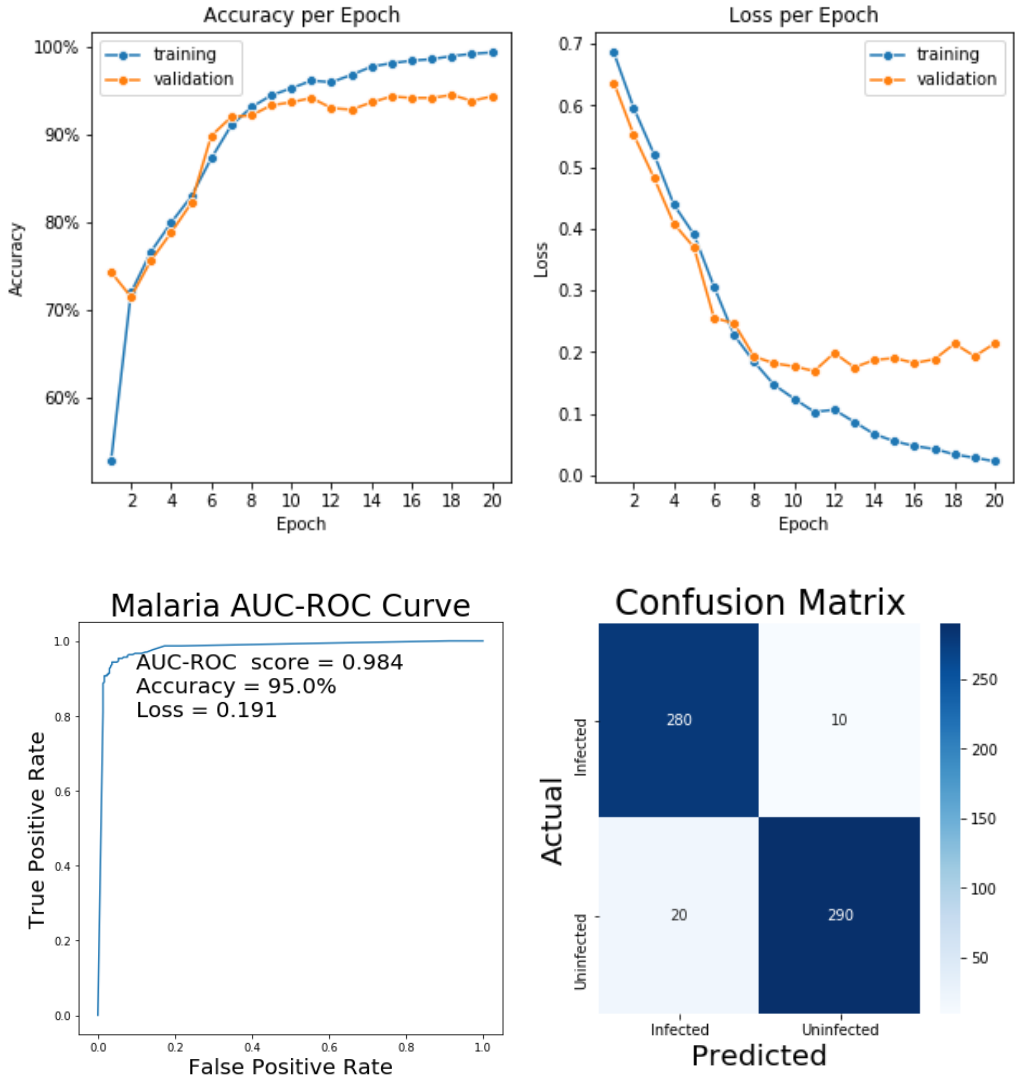
- 建立CNN模型

>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)))
)
```

Model Output

- Accuracy、Loss趨勢圖
- ROC分析
- 混淆矩陣



Adjustment parameters

改變optimizer

```
model.compile(optimizer='adagrad',
              loss='binary_crossentropy',
              metrics=[tf.keras.metrics.binary_accuracy])
model.summary()
```

accuracy → 81.2%

改變optimizer、激活函數

```
Flatten(),
Dropout(0.50),
Dense(128, activation='softmax'),
#二元分類，因此採用sigmoid
Dense(1, activation='sigmoid')

'optimizer='adagrad',
```

accuracy → 76.5%

改變optimizer、激活函數、維度

```
Flatten(),
Dropout(0.50),
Dense(64, activation='softmax'),
#二元分類，因此採用sigmoid
Dense(1, activation='sigmoid')

'optimizer='adagrad',
```

accuracy → 79.7%

Adjustment parameters

改變維度

```
    Flatten(),
    Dropout(0.50),
    Dense(64, activation='relu'),
    #二元分類，因此採用sigmoid
    Dense(1, activation='sigmoid')
])
```

accuracy → 94%

改變激活函數

```
Flatten(),
Dropout(0.50),
Dense(128, activation='softmax'),
#二元分類，因此採用sigmoid
Dense(1, activation='sigmoid')
```

accuracy → 90.5%

原Model增加一層Dense layer

```
Dropout(0.50),
Dense(128, activation='relu'),
Dense(64, activation='relu'),
#二元分類，因此採用sigmoid
Dense(1, activation='sigmoid')
```

accuracy → 94.2%

各增加一層Convolution Layer、Pooling Layer

```
Conv2D(32, (3,3), activation='relu', input_shape=(50,50, 3),padding='same'),
MaxPooling2D(2, 2),
Conv2D(32, (3,3), activation='relu'),
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),
```

accuracy → 95.8%

Adjustment parameters

再增加一層Dense layer

```
Dropout(0.50),
Dense(128, activation='relu'),
Dense(64, activation='relu'),
#二元分類，因此採用sigmoid
Dense(1, activation='sigmoid')
```

accuracy → 94.5%

batch數量往下調整

```
batch size =  200
```

accuracy → 95.7%

batch數量往上調整

```
batch size =  400
```

accuracy → 94%

再增加一層Convolution Layer 及 Pooling Layer

```
Conv2D(32, (3,3), activation='relu', input_shape=(50,50, 3),padding='same'),
MaxPooling2D(2, 2),
Conv2D(32, (3,3), activation='relu'),
MaxPooling2D(2, 2),
Conv2D(32, (3,3), activation='relu'),
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),
```

accuracy → 95%

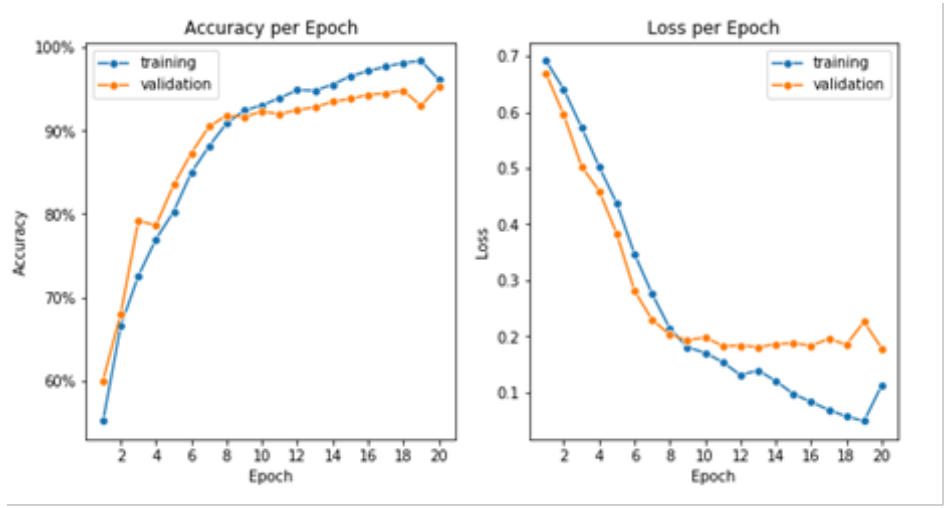
Adjustment parameters

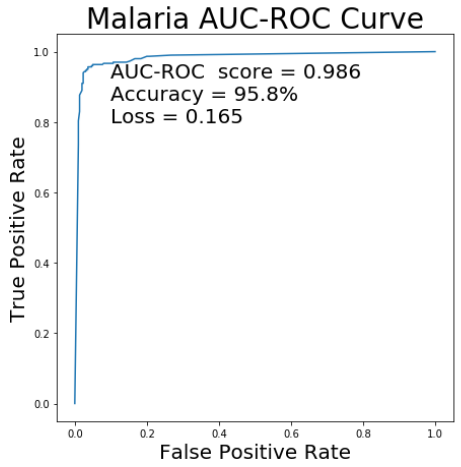
	optimizer	activation	input_dim	Convolution Layer	Pooling Layer	Dense Layer	batch	AUC-ROC Score	Loss	Accuracy
original	adam	relu	128	3層	3層	2層	300	0.984	0.191	95%
Test1	adagrad	relu	128	3層	3層	2層	300	0.908	0.401	81.2%
Test2	adagrad	softmax	128	3層	3層	2層	300	0.801	0.627	76.5%
Test3	adagrad	softmax	64	3層	3層	2層	300	0.837	0.623	79.7%
Test4	adam	softmax	128	3層	3層	2層	300	0.925	0.443	90.5%
Test5	adam	relu	64	3層	3層	2層	300	0.983	0.266	94%
Test6	adam	relu	128	3層	3層	3層	300	0.977	0.263	94.2%
Test7	adam	relu	128	4層	4層	2層	300	0.986	0.165	95.8%
Test8	adam	relu	128	4層	4層	3層	300	0.984	0.21	94.5%
Test9	adam	relu	128	4層	4層	2層	200	0.985	0.162	95.7%
Test10	adam	relu	128	4層	4層	2層	400	0.984	0.156	94%
Test11	adam	relu	128	5層	5層	2層	300	0.987	0.16	95%

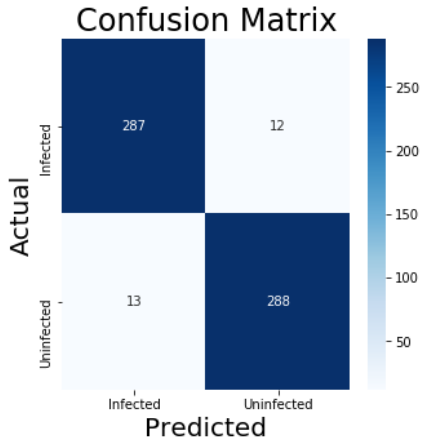
- 準確率最好為95.8%

Adjustment parameters

- Output







conclusion

- CNN模型的辨識準確率在大部分測試中均能達到90%以上
- 若要再提升準確率 → train更多圖片
  → 變更其他參數
- 由於應用於醫療方面，希望準確率能追求至99%以上