口腔鱗狀細胞癌檢測分析

Group 4

112034548 尹品玲

112034549 趙子嫣

112034543 邱愉平

背景介紹

問題背景

- 口腔癌是全球常見的惡性腫瘤之一，其中口腔鱗狀細胞癌（OSCC）是最常見的類型。
- 早期診斷、早期治療，治癒率可高達80%以上。
- 影像處理技術革新傳統病理診斷方法。

解決工具

- 使用口腔正常上皮組織圖像和口腔鱗狀細胞癌組織圖像。
- 運用深度學習方法對圖像進行特徵提取和分類。

期望成效

- 模型辨識結果可作為診所人員辨別顧客口腔狀態的參考。
- 提升診斷效率與準確性。

問題分析－5W1H

What

- 人為診斷受限於醫生經驗和判斷。
- 影響診斷準確性和可靠性。
- 人工診斷需花大量時間和精力。

Who

- 顧客
- 診所人員

When

- 顧客有口腔健康狀態疑慮時。

Where

- 牙醫診所

Why

- 及早發現與治療提升治癒率。
- 影像處理和機器學習技術革新傳統診斷方法。
- 提升診斷效率和準確性。

How

- 將網頁與AI方法結合，藉由圖像上傳系統回傳辨識結果。

資料集介紹

組織病理學圖像

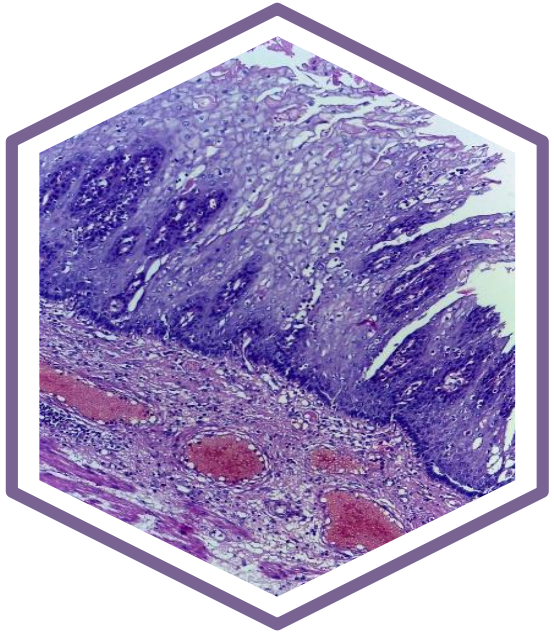
正常口腔上皮組織圖像、
口腔鱗狀細胞癌圖像

收集時間範圍

2016/10 ─ 2017/11

圖像數據來源

印度的Ayursundra Healthcare Pvt. Ltd. 和Dr B. Borooah癌症研究所（BBCI）



數據收集方式

配備相機的顯微鏡拍攝

圖像標記

收集對應的病理報告

訓練資料量

Normal 2494 /
OSCC 2698

研究方法

01

Convolutional Neural Networks, CNN

- 三種類型的層構成：卷積層（Convolutional Layer）、池化層（Pooling Layer）、全連接層（Fully Connected Layer），適用於處理具空間層次結構的數據，如圖像。
- 簡單靈活、易於實現，對於計算資源受限的環境，CNN可能更容易部署和優化。

02

EfficientNet

- Google AI在2019年提出的新型卷積神經網路架構，引入複合縮放方法（同時縮放網路的寬度、深度和輸入圖像的分辨率），以在保持模型效率的同時最大化準確性。
- 具有一系列的模型（從B0到B7），每個模型比前一個更大、更複雜。

03

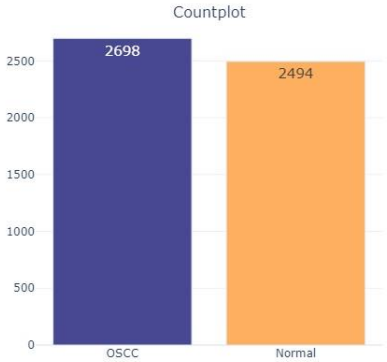
ResNet

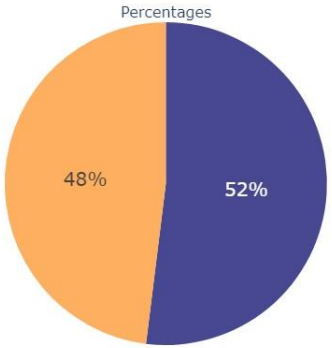
- ResNet由微軟研究院的Kaiming He等人在2015年提出，藉由引入一種創新的結構—殘差模塊（residual block）解決網路加深時常見的梯度消失問題。
- 具有多版本，差異在於層數的配置，有包含ResNet-18、ResNet-50、ResNet-101等。

資料前處理（1/3）

檢視資料分布情形

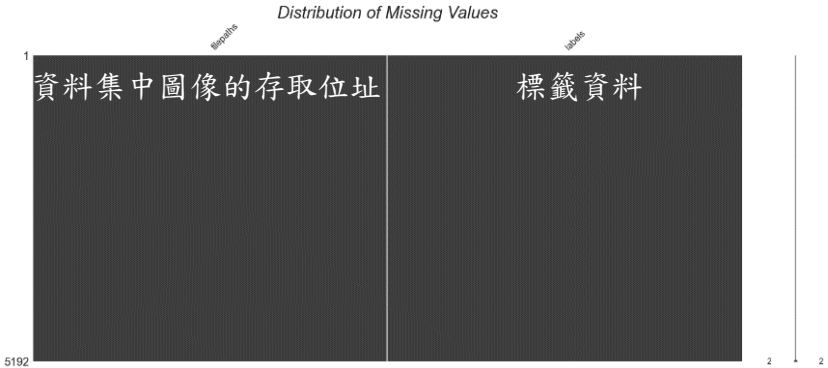
視覺化呈現，可以發現兩個類別
（Normal和OSCC）的分布為平衡分布





檢視資料缺失情形

透過Python的missingno套件了解
並無缺失值存在於數據當中



資料前處理（2/3）

資料切割

將數據切割成三部分，分別為訓練集資料佔70%、驗證集資料佔15%、測試集資料佔15%

```
# training data:70% dummy data:30%
train_df, dummy_df = train_test_split(df,  train_size= 0.7, shuffle= True, random_state= 123)

# validation data:15% testing data:15%
valid_df, test_df = train_test_split(dummy_df,  train_size= 0.5, shuffle= True, random_state= 123)
```

	Normal	OSCC
訓練集	1755	1859
驗證集	420	359
測試集	360	419

資料前處理（3/3）

數據增強

- 針對訓練集資料進行數據增強，提高模型泛化能力
- 使用Keras庫中的ImageDataGenerator
  - 隨機水平翻轉圖像（horizontal_flip=True）
  - 隨機旋轉角度（rotation_range=90）
  - 隨機水平平移（width_shift_range=0.1）

```
#traing data的數據增強
tr_gen = ImageDataGenerator(horizontal_flip=True,   #隨機水平翻轉圖像
                            rotation_range=90,      #隨機旋轉角度
                            width_shift_range=0.1   #隨機水平平移
                            )
```

實際運用

- 使用flow_from_dataframe將定義好的增強操作應用到具體數據集上
- 從Dataframe中加載標籤和圖像文件路徑
- 將數據以適合訓練的形式提供給後續訓練的模型

```
train_gen = tr_gen.flow_from_dataframe(train_df,
                                       x_col= 'filepaths',
                                       y_col= 'labels',
                                       target_size= img_size,
                                       class_mode= 'categorical',
                                       color_mode= 'rgb',
                                       shuffle= True,
                                       batch_size= batch_size)
```

模型訓練與超參數優化

- 訓練之模型：CNN、EfficientNetB3、ResNet-50
- L_9直交表進行實驗設計做超參數優化
- 比較依據：各模型各組合的Test Accuracy

	Level 1	Level 2	Level 3
Learning Rate	0.001	0.005	0.01
Optimizer	SGD	Adamax	RMSprop
Dropout Rate	0.35	0.45	0.55

CNN（1/2）

定義輸入的圖片大小、RGB彩色圖像、類別

```
#模型建立
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_gen.class_indices.keys()))

cnn_model = Sequential([
    #第一層
    Conv2D(32, (3, 3), activation='relu', input_shape=img_shape),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    #第二層
    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    #第三層
    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    #拉平
    Flatten(),
```

基層 建立三個卷

將多維的卷積特徵圖拉平成一維向量

```
    #全連接層
    Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(rate=dropoutRate),

    #輸出層
    Dense(class_count, activation='softmax')
])

cnn_model.compile(Adamax(learning_rate= learningRate), loss='categorical_crossentropy', metrics=['accuracy'])

cnn_model.summary()
```

全連接層並使用L2正則化、Dropout來避免過擬合

輸出層

設置early stopping的機制避免過擬合或浪費計算資源

```
early_stopping = EarlyStopping(monitor='val_loss',
                               patience=10,    #如果有連續10個epochs的val_loss沒有改善就會提早停止
                               restore_best_weights=True,   #訓練完成後，模型的權重會恢復到val_loss最佳的一次訓練結果
                               mode='min',    #期望val_loss越小越好
                              )


batch_size = 32
epochs = 30
```

```
history = cnn_model.fit(x=train_gen,
                        epochs= epochs,
                        verbose= 1,
                        validation_data= valid_gen,
                        validation_steps= None,
                        shuffle= True,
                        batch_size= batch_size)
```

CNN（2/2）

組合	Learning Rate	Optimizer	Dropout Rate	Test Accuracy
1	0.001	SGD	0.35	53.79%
2	0.001	Adamax	0.45	57.38%
3	0.001	RMSprop	0.55	62.90%
4	0.005	SGD	0.45	62.39%
5	0.005	Adamax	0.55	65.34%
6	0.005	RMSprop	0.35	76.89%
7	0.01	SGD	0.55	76.77%
8	0.01	Adamax	0.35	63.54%
9	0.01	RMSprop	0.45	53.79%

EfficientNetB3（1/2）

EfficientNetB3的預訓練模型

批量歸一化層，有助於加快訓練穩定模型

包含256個神經元，使用ReLU激活函數

```
#模型建立
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_gen.class_indices.keys()))

base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top= False, weights= "imagenet",
                                                               input_shape= img_shape, pooling= 'max')

efficentNet_model = Sequential([
    base_model,
    BatchNormalization(axis= -1, momentum= 0.99, epsilon= 0.001),
    Dense(256, kernel_regularizer= regularizers.l2(l= 0.016), activity_regularizer= regularizers.l1(0.006),
                bias_regularizer= regularizers.l1(0.006), activation= 'relu'),
    Dropout(rate= dropoutRate, seed= 123),
    Dense(class_count, activation= 'softmax')
])
efficentNet_model.compile(Adamax(learning_rate= learningRate), loss= 'categorical_crossentropy', metrics= ['accuracy'])

efficentNet_model.summary()
```

Dropout層

輸出層

```
early_stopping = EarlyStopping(monitor='val_loss',
                               patience=10,     #如果有連續10個epochs的val_loss沒有改善就會提早停止
                               restore_best_weights=True,   #訓練完成後，模型的權重會恢復到val_loss最佳的一次訓練結果
                               mode='min',     #期望val_loss越小越好
                              )



batch_size = 32
epochs = 30

history = efficentNet_model.fit(x=train_gen,
                    epochs= epochs,
                    verbose= 1,
                    validation_data= valid_gen,
                    validation_steps= None,
                    shuffle= True,
                    batch_size= batch_size)
```

EfficientNetB3（2/2）

組合	Learning Rate	Optimizer	Dropout Rate	Test Accuracy
1	0.001	SGD	0.35	91.27%
2	0.001	Adamax	0.45	97.18%
3	0.001	RMSprop	0.55	93.45%
4	0.005	SGD	0.45	95.51%
5	0.005	Adamax	0.55	96.53%
6	0.005	RMSprop	0.35	82.67%
7	0.01	SGD	0.55	98.20%
8	0.01	Adamax	0.35	85.11%
9	0.01	RMSprop	0.45	83.70%

ResNet-50（1/2）

ResNet-50的預訓練模型

批量歸一化層，有助於加快訓練穩定模型

包含256個神經元，使用ReLU激活函數

```
#模型建立
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_gen.class_indices.keys()))

base_model = ResNet50(include_top=False, weights="imagenet", input_shape=img_shape, pooling='max')

resNet_model = Sequential([
    base_model,
    BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001),
    Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.016),
          activity_regularizer=regularizers.l1(0.006), bias_regularizer=regularizers.l1(0.006)),
    Dropout(rate=dropoutRate, seed=123),
    Dense(class_count, activation='softmax')
])
resNet_model.compile(Adamax(learning_rate= learningRate), loss='categorical_crossentropy', metrics=['accuracy'])

resNet_model.summary()

early_stopping = EarlyStopping(monitor='val_loss',
                               patience=10,     #如果有連續10個epochs的val_loss沒有改善就會提早停止
                               restore_best_weights=True,    #訓練完成後，模型的權重會恢復到val_loss最佳的一次訓練結果
                               mode='min',     #期望val_loss越小越好
                              )


batch_size = 32
epochs = 30

history = resNet_model.fit(x=train_gen,
                   epochs= epochs,
                   verbose= 1,
                   validation_data= valid_gen,
                   validation_steps= None,
                   shuffle= True,
                   batch_size= batch_size)
```

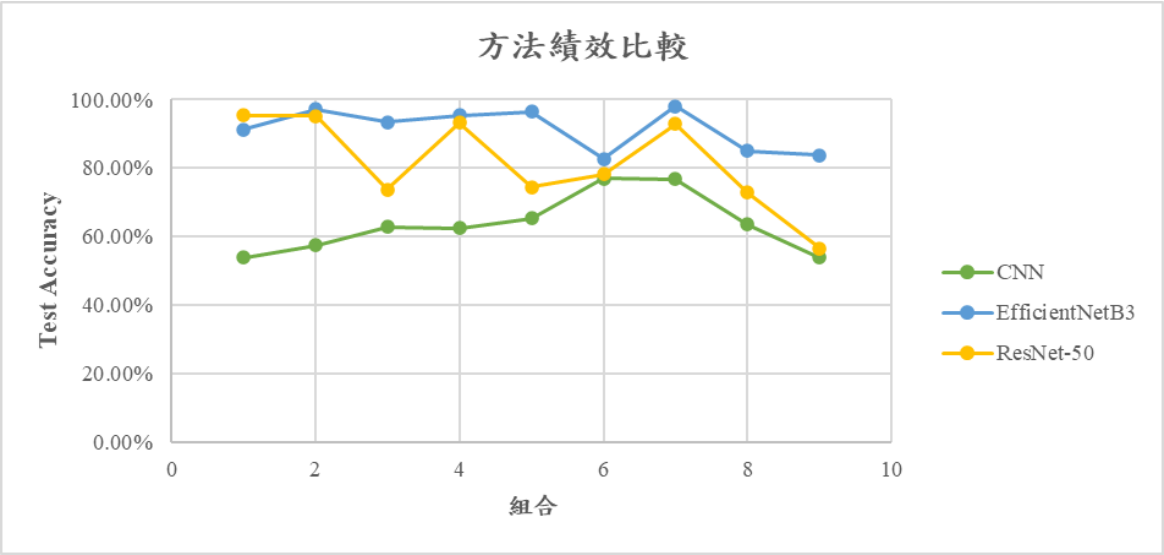
Dropout層

輸出層

ResNet-50（2/2）

組合	Learning Rate	Optimizer	Dropout Rate	Test Accuracy
1	0.001	SGD	0.35	95.38%
2	0.001	Adamax	0.45	95.25%
3	0.001	RMSprop	0.55	73.68%
4	0.005	SGD	0.45	93.20%
5	0.005	Adamax	0.55	74.45%
6	0.005	RMSprop	0.35	78.18%
7	0.01	SGD	0.55	92.94%
8	0.01	Adamax	0.35	72.91%
9	0.01	RMSprop	0.45	56.48%

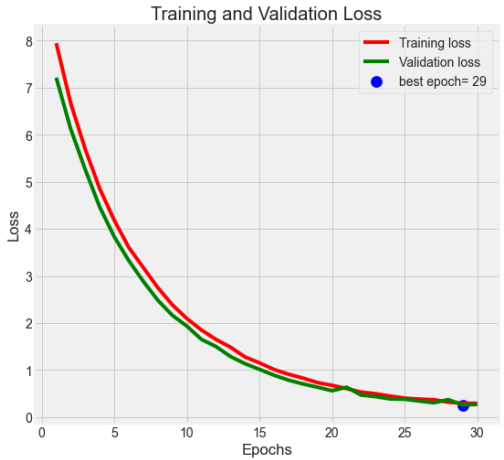
方法績效比較（1/2）

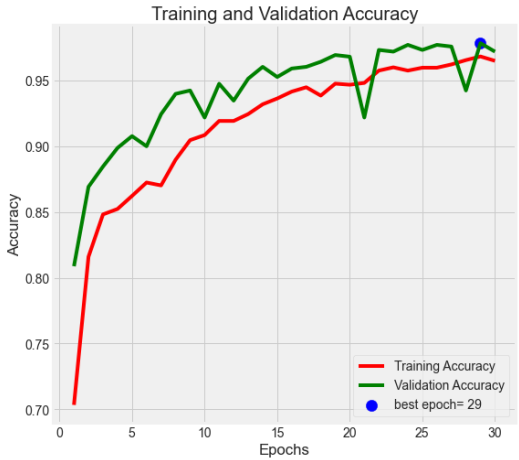


Model	Learning Rate	Optimizer	Dropout Rate	Test Accuracy
CNN	0.005	RMSprop	0.35	76.89%
EfficientNetB3	0.01	SGD	0.55	98.20%
ResNet-50	0.001	SGD	0.35	95.38%

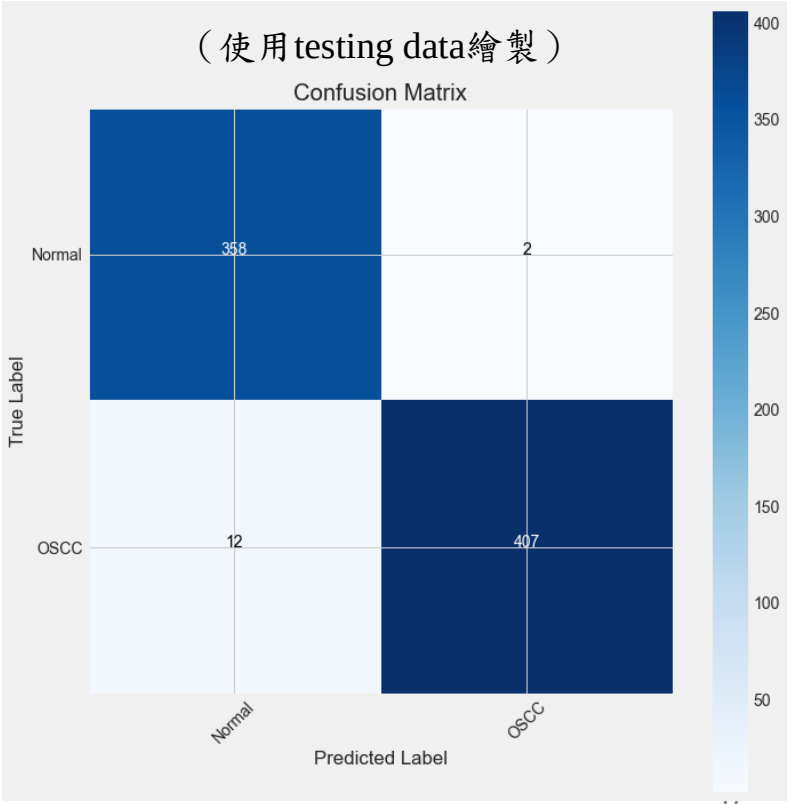
方法績效比較（2/2）

EfficientNetB3 / 0.01 / SGD / 0.55

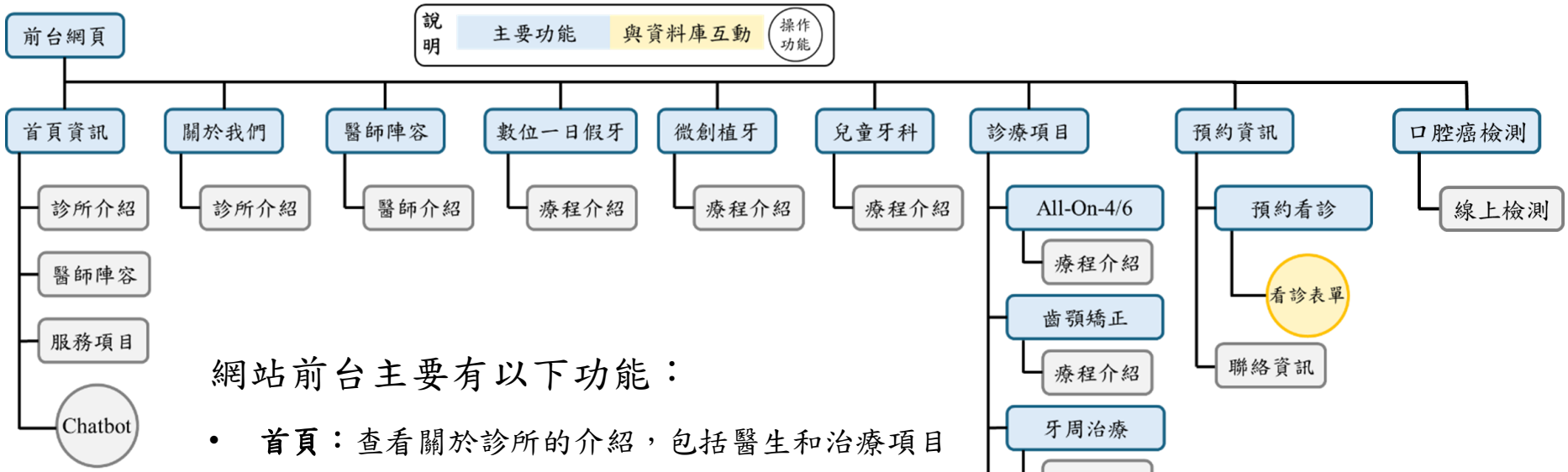




（使用testing data繪製）



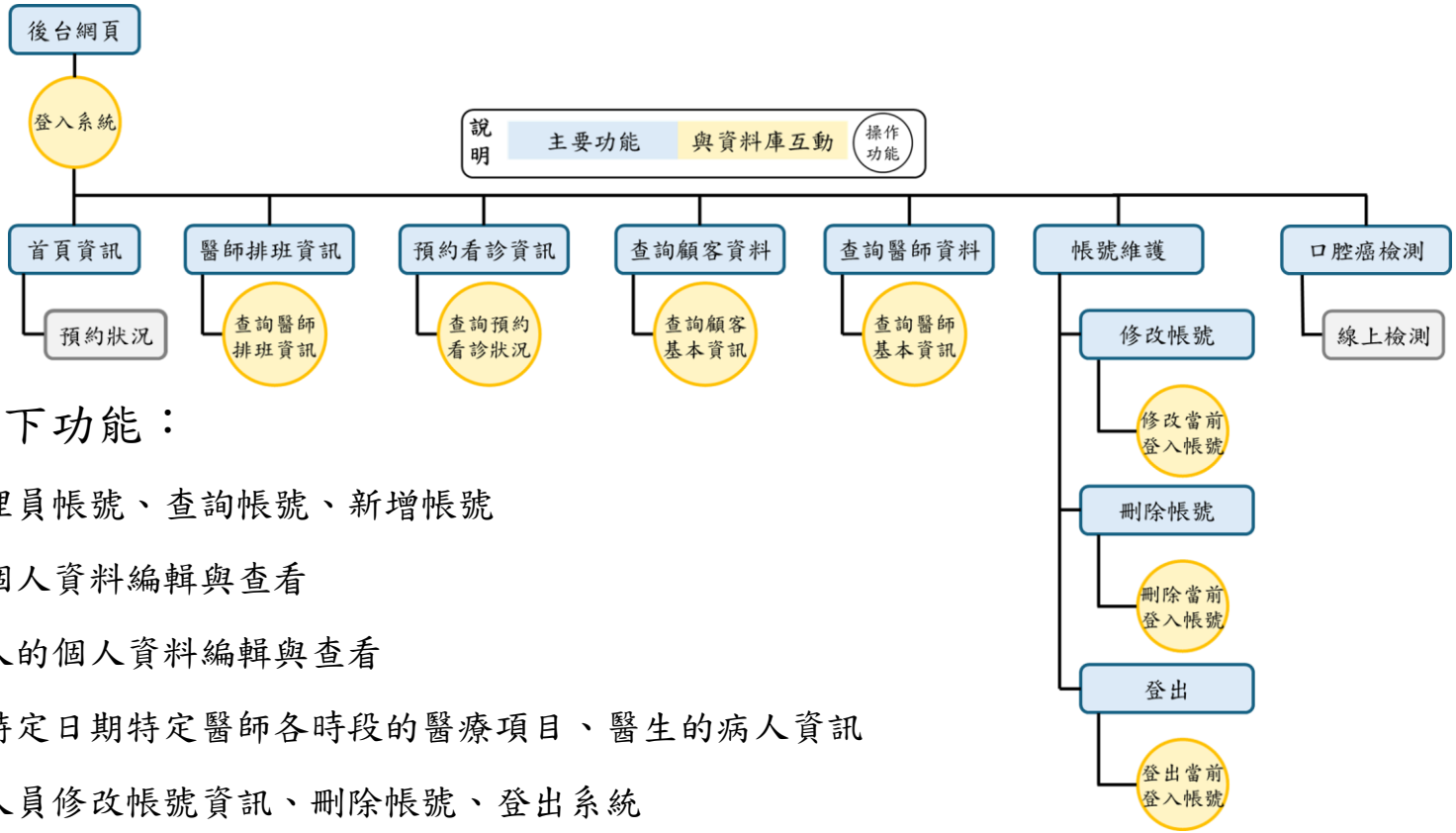
前台網頁設計架構



網站前台主要有以下功能：

- **首頁**：查看關於診所的介紹，包括醫生和治療項目
- **醫師陣容**：介紹各醫生的專業診療項目
- **各式醫療項目**：診所內的醫療項目介紹
- **預約資訊**：讓顧客輸入資料後進行預約作業
- **口腔癌檢測**：顧客可自行上傳圖片做檢測

後台網頁設計架構



網站後台主要有以下功能：

- **首頁登入**：登入管理員帳號、查詢帳號、新增帳號
- **醫生資訊**：醫生的個人資料編輯與查看
- **顧客資訊**：預約客人的個人資料編輯與查看
- **約診資訊**：可查詢特定日期特定醫師各時段的醫療項目、醫生的病人資訊
- **帳號維護**：讓管理人員修改帳號資訊、刪除帳號、登出系統
- **口腔癌檢測**：管理人員可上傳圖片檢測並記錄

網頁結合口腔癌檢測（1/2）

Flask是使用Python語言編寫的Web應用框架

使用者透過POST協定從前端網頁傳送要預測的圖片 → 後端程式收到資料後會送給事先打包好的模型 → 將模型預測結果回傳到前端使用者

實現上述功能主要需三部分：

1. 將預訓練好的模型儲存成.h5的檔案

```
model.save('model.h5')
```

2. 透過Flask建立API，並載入預訓練好的模型

```
from flask import Flask, request, render_template
import numpy as np
from tensorflow.keras.models import load_model
import traceback
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.efficientnet import preprocess_input
from werkzeug.utils import secure_filename
import os

app = Flask(__name__, static_folder='static', static_url_path='/static')


#輸入預先訓練好模型
model = load_model('model.h5')
```

網頁結合口腔癌檢測（2/2）

3 預測功能實踐

- 建立GET的路由@app.route(‘/’)，單引號中的內容代表使用者呼叫API的路徑位置。
- @app.route(‘/predict’, methods=[‘POST’])負責接收使用者上傳的圖片資料。
- app.run()將此API部署在伺服器的3000 PORT中，供使用者使用。

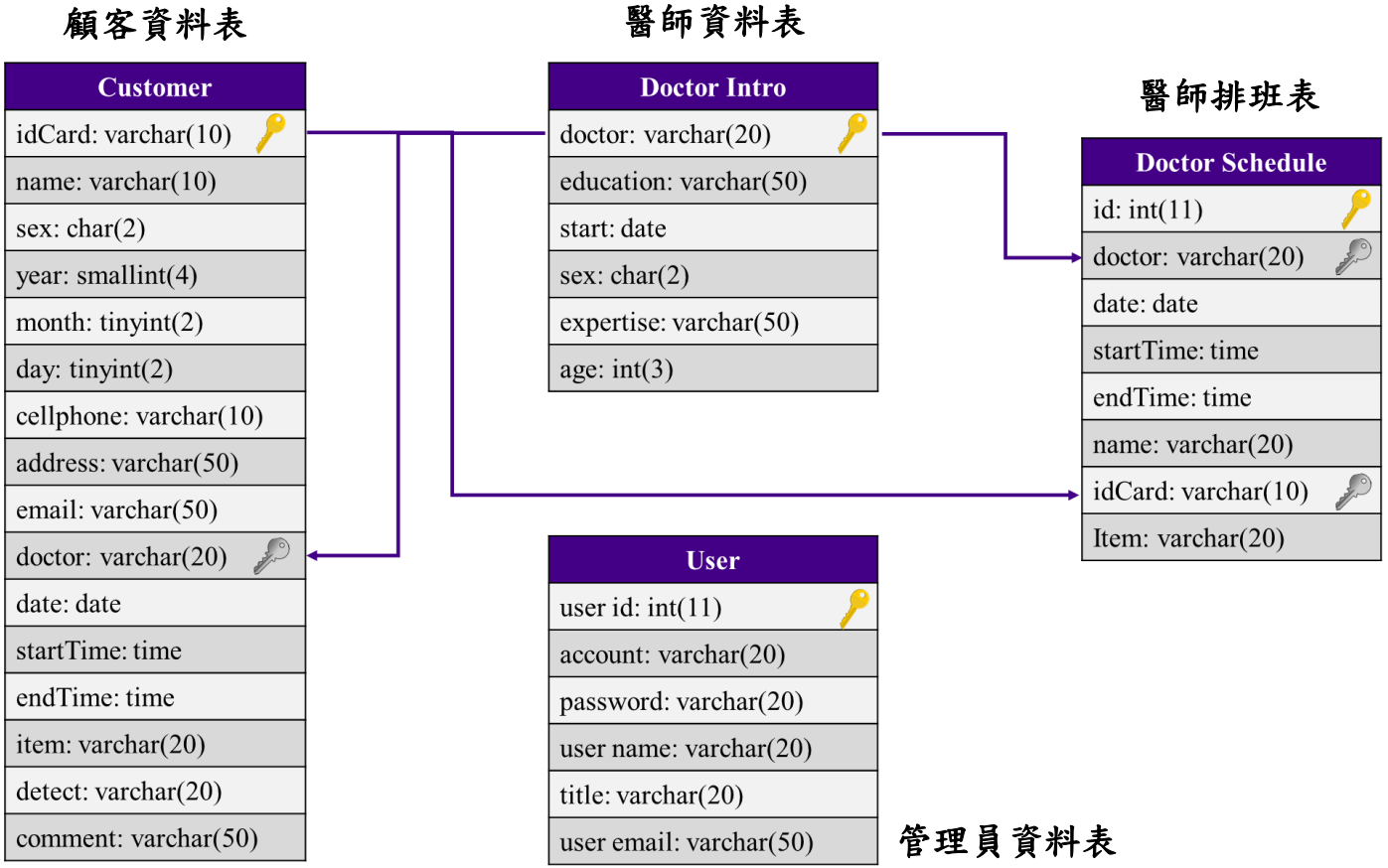
```python
@app.route('/')
def home():
    return render_template("口腔癌檢測.html")

@app.route('/predict', methods=['POST'])
def predict():
    if img_file:
        filename = secure_filename(img_file.filename)
        uploads_dir = 'uploads'
        os.makedirs(uploads_dir, exist_ok=True)
        save_path = os.path.join(uploads_dir, filename)
        img_file.save(save_path)
        class_labels = ['Normal', 'Oral Cancer']

        img = image.load_img(save_path, target_size=(224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = preprocess_input(img_array)

        #進行預測
        prediction = model.predict(img_array)
        predicted_class_index = np.argmax(prediction)

        predicted_class_label = class_labels[predicted_class_index]

        return render_template("口腔癌檢測.html", prediction_display_area="預測結果：{}".format(predicted_class_label), predic
    return render_template("口腔癌檢測.html", prediction_display_area="An error occurred", prediction_text_color="color: red;

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=3000, debug=False)
```

網頁Demo

前台網頁

後台網頁

Entity-Relationship Model (E-R Model)



結論與未來展望

為顧客創造客製化治療服務

針對診斷出有口腔癌的病人提供個性化療程，並提醒回診時間

將模型與網頁結合，使用者可直接上傳圖像並獲得預測結果

成功訓練機器學習模型用於識別口腔鱗狀細胞癌

THANKS!