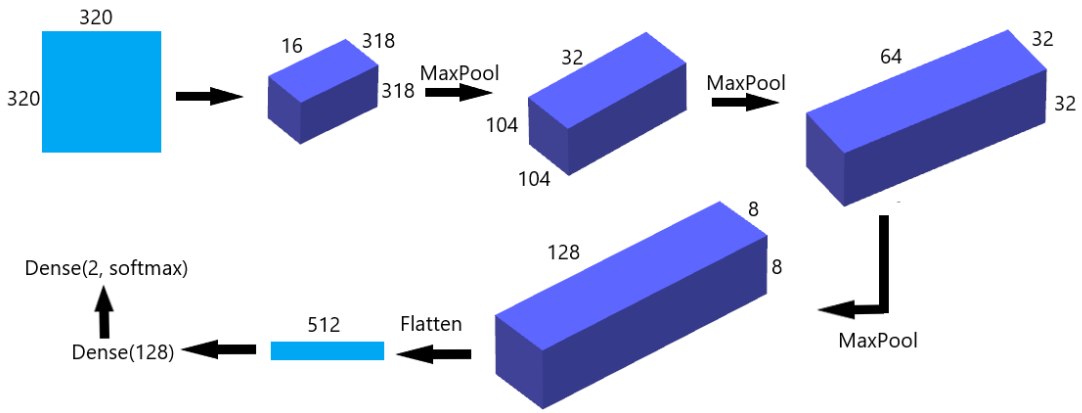
ML_assignment3

本次作業主要嘗試了兩種方法，一是自己手刻 model 出來，二是利用 pre-trained model，如 VCC16、resnet_v2、Inception2 等等。

Method1:手刻 model

```
model = Sequential()
# convolutional layer
model.add(Conv2D(16, kernel_size=(3,3), strides=(1,1),
                 padding='valid', activation='relu', input_shape=(320,320, 1)))
model.add(MaxPool2D(pool_size=(3, 3)))
model.add(Conv2D(32, kernel_size=(3,3), padding='valid', activation='relu'))
model.add(MaxPool2D(pool_size=(3, 3)))
model.add(Conv2D(64, kernel_size=(3,3), padding='valid', activation='relu'))
model.add(MaxPool2D(pool_size=(3, 3)))
model.add(Conv2D(128, kernel_size=(3,3), padding='valid', activation='relu'))
model.add(MaxPool2D(pool_size=(3, 3)))
# flatten output of conv
model.add(Flatten())
# hidden layer
model.add(Dense(128, activation='relu'))
# output layer
model.add(Dense(2, activation='softmax'))
```



Model Summary:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 318, 318, 16)      160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 106, 106, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 104, 104, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 34, 34, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 128)               65664
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 163,074
Trainable params: 163,074
Non-trainable params: 0
```

自己接的效果並不怎麼好，將 train data 以 7:3 的比例分為 train, test，設定 batch_size = 16、epoch = 20 進行訓練，最後得到的 F1 score 大約為 0.3 上下。

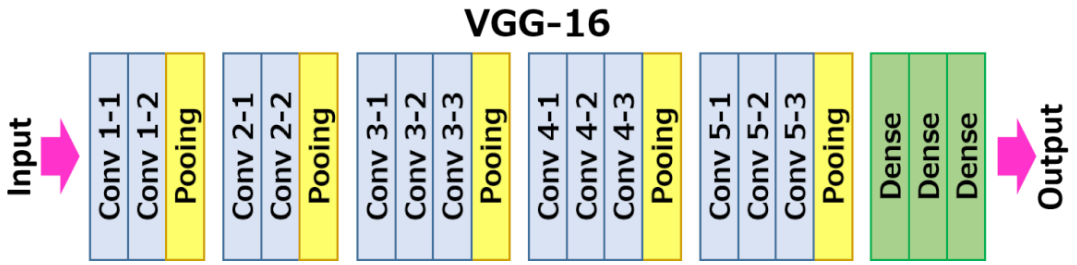
```
TN: 319 , FN: 34 , TP: 17 , FP: 48
precision: 0.26153846153846155 , recall: 0.3333333333333333
F1: 0.29310344827586204
```

個人認為結果不好的原因大約有幾點：

1. 自己接的 Model 無法有效地抓取到 features。
2. Data 的 size 太小，造成訓練效果不佳

Method2:搭配 pre-trained model 使用(Transfer-Learning)

搭配 pre-trained model 使用。利用較有名的 pre-trained model，例如：VGG16, ResNet, InceptionV3 等，將 data 丟入 pre-trained model 後，其輸出再接上自己設定的 layer。以 VGG16 為例：




```
base_model = VGG16(input_shape = (320, 320, 3), # Shape of our images
                   include_top = False, # Leave out the last fully connected layer
                   weights = 'imagenet')

for layer in base_model.layers:
    layer.trainable = False

# Flatten the output layer to 1 dimension
x = Flatten()(base_model.output)

# Add a fully connected layer with 512 hidden units and ReLU activation
x = Dense(2048, activation='relu')(x)
x = Dense(1024, activation='relu')(x)

# Add a final sigmoid layer for classification
x = Dense(2, activation='softmax')(x)
```

Base_model 為已經 pre-trained 好的 model，因此將其 trainable 設定成 false，單純訓練在 pre-trained 之後我們所接的 Dense(2048)、Dense(1024)、Dense(2)。

使用 pre-trained model 時遇到的問題為，大部分的 pre-trained model 都是以彩色相片進行訓練，與這次作業使用的 grayscale image 不同，RGB image 的 shape 為(, , 3)，也就是有 3 個 channels，為了解決此問題，在網路上查到的方法是，將 grayscale 複製成 3 份，進而當作有 3 個 channels。

Data 以與 method1 一樣的方式進行分割，最後訓練出來的 model 的 performance 也大約為 0.3。

Data augmentation:

為了讓 model 更加穩定，並且增加 data 的數量，嘗試將 image 進行旋轉、翻轉、縮放等方式，但實際測試下來感覺並沒有什麼效果。

Ex. Image Rotation

```
for i in range(len(X_train)):
        rotate_X_train[i] = rotate(X_train[i], angle=randint(-90, 90), reshape=False)
```

複製一組與原 data 相同的 image，將其進行-90 度~90 度的隨機翻轉，再將翻轉過後的 images 加入到 data 中進行訓練。

Bonus

先利用 assignment3 所做出來的 model，以 images 作為 Input data，predict 出患者的 outcomes，然後將 outcomes 當作 HER 中的其中一項 attribute，也就是將 CNN model 的輸出當作 40 幾樣 attributes 中的其中一樣，再用這些 attributes 來對當初使用的 Random Forest 做訓練，以得到更好的結果。

當初在做 assignment2 時，self-test 的 F1 大約都為 0.5 多一點，最後實際的 F1 為 0.51，算是 self-test 跟實際 test 的結果蠻相當的。

利用 CXR 搭配 EHR 進行訓練，在 self-test 時大約可以達到 0.7 的 F1 score。比起單純 EHR 多了快 0.2。