HW3 Report

由於本次作業，主要需實踐 Image Prediction，預測病人是否會過世。因此，本次嘗試 `CNN` 作為解題方向。

Implementation

Data Preprocessing

由於本次的 input data 為 image format，在逐一讀入 image 後，便進行了以下步驟：

1. **Train-Test Split：** 不論在有 Bonus 或沒有 Bonus 的版本中，我皆使用 `trainTestSplit` 進行資料拆分，以確保 Training Set 與 Validation Set 的資料亂度。
2. **取出 Dead Person Data 進行增生**：因為原始 Data 有 `981` 筆 Alive Person 與 `133` 筆 Dead Person，uneven 的情況下會使 training result 較差，因此這部分有進行兩個 `augmentation` 的操作。
    - Rescale to 64×64: 仿照 CIFAR-10 的資料格式，我試圖降低影像的大小，以求加快 training 速度。
    - Balance Alive / Dead Dataset: 首先，我取出 `Dead Person` 的 images，加進 `ImageDataGenerator` ，逐量進行影像微調（旋轉、放大、位移...），使 Alive Person 與 Dead Person 的 image 數量趨近。
    - Double 全部 Dataset: 當 Alive 與 Dead Person 的 image dataset 數量趨近時，便集體增加兩倍，以確保 CNN 能學得更多的胸腔特徵值。此操作可有效提升最終 f1 score。

    同時，在增生資料過程中，我發現需避免**變異量不得設太高，並趨近原始 Dataset**。在 Augmentation 的操作下，需確保變量的數值和 dataset 相似。例如說，當我們觀察 dataset 中的每張圖片，彼此間差異的旋轉幅度介於 10 度內時， `ImageDataGenerator` 的 `rotation_range` 便須設成 `10` ，才能得到較好的 Performance 。

```
datagen = ImageDataGenerator(
  zca_whitening=False,
```

```
  rotation_range=5,
  width_shift_range=0.2,
  height_shift_range=0.2,
  shear_range=0.,
  zoom_range=0.05,
  horizontal_flip=False,
  fill_mode='nearest'
);
```

Model Training

Convolutional Neural Network

本次使用 Keras ，建造一個 CNN Model 。在嘗試過幾種 Conv \ Filtering \ Max Pooling 的搭配後，得出以下設計：

```
model.add(layers.Conv2D(32, (7, 7), strides=2, input_shape=(64, 64, 1), activatio
n='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.BatchNormalization())

model.add(layers.Conv2D(32, (3, 3), strides=2, activation='relu'))
model.add(layers.BatchNormalization())

model.add(layers.Conv2D(32, (3, 3), strides=2, activation='relu'))
model.add(layers.BatchNormalization())

model.add(layers.Flatten())
model.add(layers.Dense(units=128, activation="relu"))
model.add(layers.Dense(units=1, activation="sigmoid"))
```

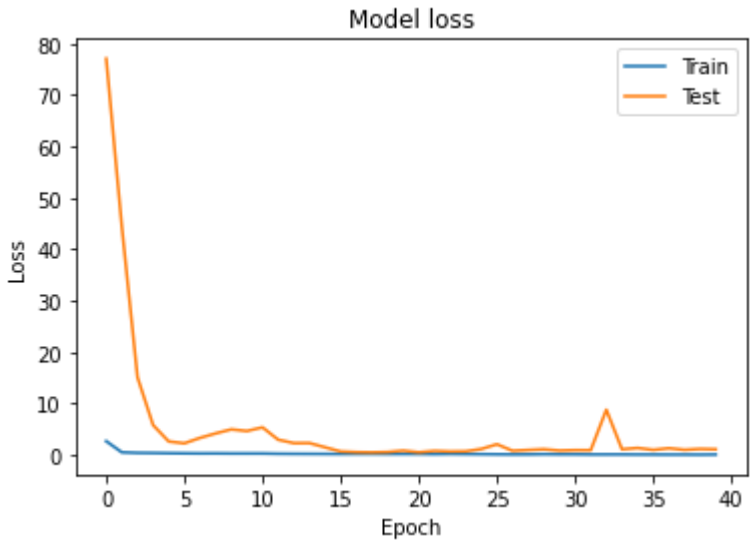
以下為詳細解說設計原因：

1. 鑑於資料，本次僅設計三大層 Network (Conv ＋ MaxPooling + BatchNormalization)，原因為：
    - 由於 Model Data 量不大，設計上以 `簡單` （避免過多層）為目標，以避免 Model Overfitting 。
    - 降低 training 的 parameters，以提高 training 效率。
    - input 僅 64×64，逐步推導後認為僅需三層架構，就可有效限縮特徵值。
    - 由於最終輸出為 Binary Output ，我發現若未定期 Normalization，Output 會趨近於 `0` ，導致 sigmoid function 皆輸出 0 ，且 F1 Score 極低。也因此，我導入 `BatchNormalization` ，使每個 layer output 能維持在一定比例內。

- 由於 MaxPooling 可能降低特徵值判斷，我在第二大層與大三大層選擇捨棄。

2. Activation Function 使用 `relu` 作為核心使用。同時，因為最終輸出以 Binary Output 為主要模式，Model 以 `binary_crossentropy` ，作為 `loss` 計算標準。
3. 因為原始 Model Training loss 很低，但 Validation loss 偏高，我使用過 `Dropout` 與 L1 與 L2 Regularization Term ，但都沒有得到太好效果。個人認為，因為 64×64 image 的 training 架構下，若再新增 Dropout ，可能讓重要特徵值被捨棄掉，進而導致最終結果較偏差。
4. 避免使用 learning rate exponential decay ，因為會導致 training loss 在 training 過程中，出現 anomaly 上升情況。實作上，我改使用 `ReduceLROnPlateau` ，讓其得以動態監控每次 `epoch` 的結果。若 validation loss 已經接近 convex 最低點時，調低 learning rates 以求最佳值。
5. 用 `Adam` 作為 `optimizers` ，對比於 `SGD` ，效果相當好。
6. 發現 Data Preprocessing ，與最終結果有絕對關係。未使用 Data Augmentation 情況下， F1 Score 大多環繞在 20 左右。套用 Data Augmentation 後，可提升至 40 左右。
7. 由於 CNN training 過程中，很容易發生 overfitting 的狀況。我套用 `ModelCheckpoint` ，記錄下每個 epoch 的 model 結果，以便擷取最佳的 epoch weight。

本次花費相當大的心力，在 fine-tuning CNN 的參數，得出最好的 f1 Score 為 `0.44`



Implementation - Bonus

在 Bonus 部分，須整合上次作業的 dataset 一同 training 。也因此，必須將 EHR 的 training 以 NN 的方式處理，最終再和 CNN image classification 合併串接。所以以下將依序講解： EHR Preprocessing \ EHR NN Training \ EHR + HW3 Combination 。

Data Preprocessing - Bonus Part for EHR Dataset

針對 **EHR** 的資料，處理了以下步驟：

1. `ed_diagnosis` 欄位進行數字化處理：由於 `ed_diagnosis` 紀錄了病人的病徵，但是以 String 形式紀錄，並無法傳入 Model 進行 Prediction 。

   為了使用 NN 的方式 train ，我用 One-Hot Encoder 的方式，將該欄位擴展至 n 個欄位 (n = total number of symptom) 。相較於 Label Encoding ，此方法可避免讓 Model ，認為數字有關聯的錯覺，也會使 NN Training 較易找出最佳值。

2. `Columns Drop` ： `admission_datetime` \ `PATIENT ID` 兩者皆與病人是否過世，理論上無太直接關聯。`sex` 經過上次 hw2 NN 測試、與 xgboost 的內建 optimizer 驗證後，也發現無太大關聯，本次實作將以上 columns Drop。

3. `Fill NAN` ：這次有相當多的數值，都產生 NaN 的數值。 `phmx` 開頭的資料中，大部分都是 `binary` 的數值，我使用 `mode` 進行資料補足。而對於非 `phmx` 開頭的資料，大多都是連續資料，就以 `mean` 進行資料補足。

   ```
   headerList = df.columns.values.tolist()

   # need to fill mode
   filteredList = list(filter(lambda name: 'pmhx' in name, headerList))

   # need to fill mean
   otherList = list(filter(lambda name: 'pmhx' not in name, headerList))

   for filteredCol in filteredList:
     df[filteredCol].fillna(df[filteredCol].mode()[0], inplace=True)

   for otherCol in otherList:
     df[otherCol].fillna(df[otherCol].mean(), inplace=True)
   ```

4. 切分 Training Data 與 Testing Data: 使用 `train_test_split` 進行切分，比率設定為 `0.2` 。使用 `train_test_split` 的原因，在於其可以 randomly 切分資料，避免 testing data 都集中在某一區塊。

5. Data Augmentation: 由於 Dead 的資料遠少於 Alive 的資料，在我完成 training / testing 的 data split 後，便開始進行 training data 的 augmentation ，試圖

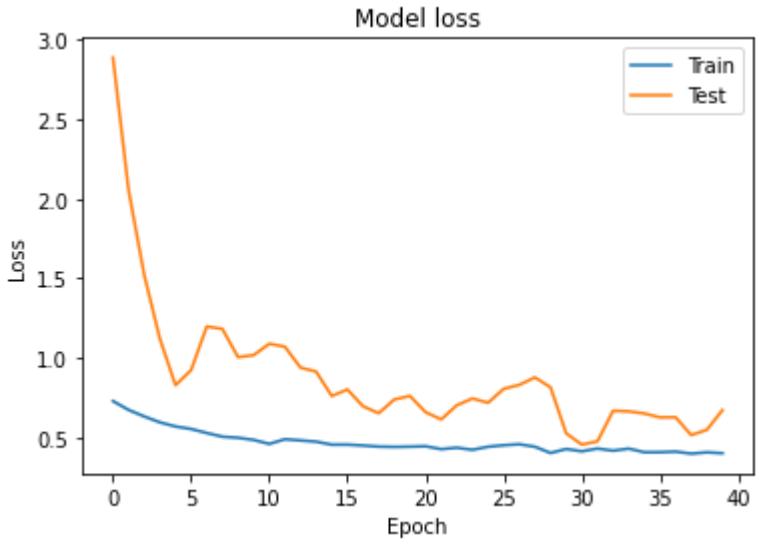
讓 training data dead 與 alive 的數量儘量平均。此方法會讓 NN 表現大幅提升。

```
from imblearn.over_sampling import SMOTE
from collections import Counter
smote = SMOTE(ratio='minority')
training_data_list, training_ans_list = smote.fit_sample(training_data_list,
 training_ans_list)
```

EHR Dataset Neural Network

本次使用 Keras，建造一個基礎的 Neural Network 。由於本次 Model Data 量不大，設計上也以 `簡單` 為目標。在試過將幾種版本後，得出以下設計：

1. 鑑於資料，本次僅設計三層 Network ，每一層的 Neural 數目不要太高，有較佳效果。
2. Activation Function 根據上課講解，以 `relu` 作為核心使用。試過 `sigmoid` 效果非常差。
3. 由於原先的資料並非介於 0 → 1 的數值，因此在 NN 中加入了 `BatchNormalization` 。
4. 因為 Testing Loss 一直降不下來，所以有使用 `Dropout` ，試圖拉低 Testing Loss 。
5. 因為最終結果是以 Binary Output ，故 Model 以 `binary_crossentropy` ，作為 `loss` 計算標準。
6. 用 `Adam` 作為 `optimizers` ，對比於 `SGD` ，效果相當好。
7. 因為 Deep Learning Workload ，越後面的 Neural 的 Output 會影響越大，有使用 Learning Rate ExponentialDecay ，試圖降低此效應。
8. Data Preprocessing 的好壞（如 label encoding 或是 one-hot 的設計、Data Augmentation），大幅影響了後期 Neural Network 的效能。
9. Batch Size 設定在 300 上下，Epoch 設定在 30 - 40 ，效果較好。



CXR **+ EHR Model Combination**

我調用 Keras `concatenate` Layer，將 **EHR** Sequence NN Model 與 CXR Sequence CNN Model，在最終進行合併，並共同輸出 Dense Layer Output 。

```
img_data = keras.layers.Input(shape=(64, 64, 1))
num_data = keras.layers.Input(shape=(49))

# image data extraction
x = layers.Conv2D(32, (7, 7), strides=2, input_shape=(64, 64, 1), activation='rel
u')(img_data)
...
x = keras.Model(inputs=img_data, outputs=x)

# numeric data extraction
y = layers.Dense(40, activation='relu', input_shape=(49,))(num_data)
...
y = keras.Model(inputs=num_data, outputs=y)

# fully connected layer
combined = keras.layers.concatenate([x.output, y.output])
z = keras.layers.Dense(units=128, activation="relu")(combined)
z = keras.layers.Dense(units=1, activation="sigmoid")(z)
model = keras.Model(inputs=[x.input, y.input], outputs=z)
```

也因此，我可以一起傳送 EHR Data 與 CXR Data 進行 Model Training ，最終便可得到 binary classification 的結果。

How to use the model

打開 `106062322_HW3_Model.ipynb` & `Bonus_106062322_HW3_Model.ipynb`後，直接執行 `Predict Final Result Block` ，即可完成 Predict ，得到 final csv。