網路安全的資料科學 期末報告　　資管三 106306062 郭宇雋

一、題目

Malware Detection with Static Analysis and Model Ensemble

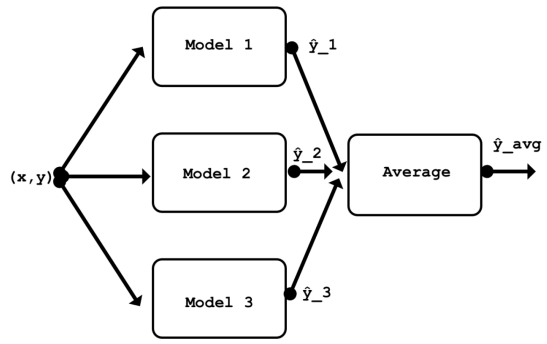
二、動機與想解決的問題

對於 Malware Detection，在課堂上有學到很多不同的方法來做 Malware Family 的分類，包括使用 PE file 的 Header Data，或是透過 Static Analysis 的方法去分析該程式使用了哪些 Library，或是透過 Dynamic Analysis 的方式去分析程式執行期間有做哪些 API Call，甚至還可以把 PE file 中每個 byte 的內容轉換成圖像，進而透過 CNN 去做影像分析來分類 Malware。
因此，我想研究的是，若是對於同一支程式，我們不單單只用其中一種分析方式來檢測，而是集合上述各種分析方式，先用適當的模型訓練不同特徵，最後再透過 Neural Network 的 Dense Layer 合併這些模型的預測結果，如此一來是否會得到更高的準確率。

三、文獻參照

1) Title: Ensembling ConvNets using Keras
Ref: https://towardsdatascience.com/ensembling-convnets-using-keras-237d429157eb

本文作者使用了三個模型集成一個最終模型，而其結果顯示，集成模型的準確率的確相較單一模型有所提升。
然而，此文獻的三個模型使用的是相同的 Input，且在模型集成的部分使用的是簡單平均法。

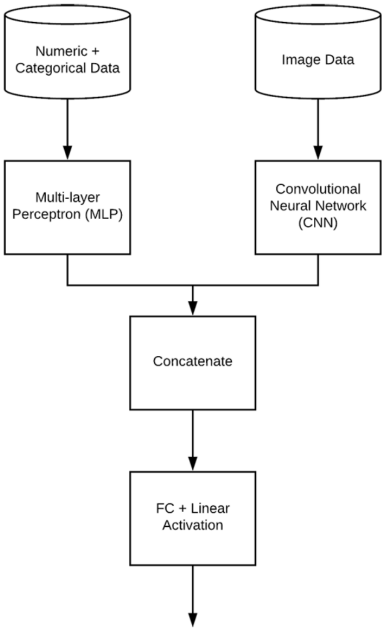


2) Title: Keras: Multiple Inputs and Mixed Data
Ref: https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/

本文作者在模型集成上選擇了另一種方法，他將不同的資料分別餵進兩個不同的模型，而後透過 Concatenate Layer 將兩個模型各自的結果合併，最後再接上一層 Fully Connected Layer (Dense Layer)，來得到最終新的預測結果。而這篇文獻的方法也是我在此報告中欲採用的做法。

此文獻的最終結果，集成模型的 MAPE 為 27.52%，
而個別的 MLP (DNN) 模型的 MAPE 為 22.71%，
而個別的 CNN 模型的 MAPE 為56.91%。

可見此文獻的結果顯示，該集成模型的準確率雖然很不錯，
然而卻沒有比單獨一個 MLP (DNN) 模型的確率來得好，
因此透過此份報告的資料集與研究過程，我想知道的是，
如果將此建構模型的概念，套用到此份報告的資料集
(Malware Detection) 上，也就是用這樣的 Model
Ensemble 概念來建構 Malware Detection System，如此
的結果會不會比訓練單一模型的預測結果來得準確。



四、資料集參照與介紹

[1] Angelo Oliveira, "Malware Analysis Datasets: PE Section Headers", IEEE Dataport, 2019. [Online]. Available: http://dx.doi.org/10.21227/2czh-es14. Accessed: Jun. 13, 2020.
[2] Angelo Oliveira, "Malware Analysis Datasets: Top-1000 PE Imports", IEEE Dataport, 2019. [Online]. Available: http://dx.doi.org/10.21227/004e-v304. Accessed: Jun. 13, 2020.
[3] Angelo Oliveira, "Malware Analysis Datasets: Raw PE as Image", IEEE Dataport, 2019. [Online]. Available: http://dx.doi.org/10.21227/8brp-j220. Accessed: Jun. 13, 2020.

- PE Section Headers 資料集為 Static Analysis Data，內容為 .text 與 .code 的 Section Header，欄位包括 hash, size_of_data, virtual_address, entropy, virtual_size。而目標欄位為 malware，此欄位為 Binary 變數 (0: Benign / 1: Malware)。
- Top-1000 PE Imports 資料集亦為 Static Analysis Data，共有 1000 個欄位，分別表示 PE file 最常 import 的 1000 個 Library，且為 Binary 變數 (0: not import / 1: import)。而目標欄位亦為 malware，此欄位為 Binary 變數 (0: Benign / 1: Malware)。
- Raw PE as Image 資料集亦為 Static Analysis Data，此資料集將 PE file 的 byte 內容轉換為 32*32 的灰階圖片，每個 pixel 的數值皆介於 0~255，故共有 1024 個欄位。而目標欄位亦為 malware，此欄位為 Binary 變數 (0: Benign / 1: Malware)。

- PE malware examples were downloaded from virusshare.com. PE benign examples were downloaded from portableapps.com and from Windows 7 x86 directories.

五、資料處理與探勘

1) 因為目標為 Model Ensemble，因此先提取出在三個資料集中都有出現的樣本，並合併這三個 Dataframe 為 train_df，以方便後續的分析。(以 hash 做為合併的 key)

- PE Section Headers Dataframe

	hash	size_of_data	virtual_address	entropy	virtual_size	malware
0	071e8c3f8922e186e57548cd4c703a5d	443392	4096	6.529624	442984	1
1	33f8e6d08a6aae939f25a8e0d63dd523	331264	4096	6.604314	330784	1
2	b68abd064e975e1c6d5f25e748663076	74240	4096	6.046789	73819	1
3	72049be7bd30ea61297ea624ae198067	219648	4096	6.497018	219524	1
4	c9b3700a77facf29172f32df6bc77f48	262144	4096	6.638142	261943	1

- Top-1000 PE Imports Dataframe

	hash	GetProcAddress	ExitProcess	WriteFile	GetLastError	CloseHandle
0	071e8c3f8922e186e57548cd4c703a5d	1	1	1	1	1
1	33f8e6d08a6aae939f25a8e0d63dd523	1	1	1	1	1
2	b68abd064e975e1c6d5f25e748663076	1	1	1	1	1
3	72049be7bd30ea61297ea624ae198067	1	1	1	1	0
4	c9b3700a77facf29172f32df6bc77f48	1	1	1	1	1

5 rows × 1002 columns

- Raw PE as Image Dataframe

	hash	pix_0	pix_1	pix_2	pix_3	pix_4	pix_5	pix_6
0	b324140e1fb35dc6b694879ba1f2be45	15	15	239	15	223	36	102
1	1d32b1326a524b163eb74af645cd34d5	234	196	8	20	182	56	27
2	e44fea4913fc9fd91b8b07c4670aeac4	196	255	5	97	35	112	219
3	95badb16d862ba94ae85c44e4b31d749	232	252	183	39	51	1	255
4	f30f32a4f42678ef49a543356d580232	81	84	204	228	255	157	76

5 rows × 1026 columns

- Merge Dataframe (which is called train_df)

	hash	size_of_data	virtual_address	entropy	virtual_size	GetProcAddress	ExitProcess
0	071e8c3f8922e186e57548cd4c703a5d	443392	4096	6.529624	442984	1	1
1	33f8e6d08a6aae939f25a8e0d63dd523	331264	4096	6.604314	330784	1	1
2	b68abd064e975e1c6d5f25e748663076	74240	4096	6.046789	73819	1	1
3	72049be7bd30ea61297ea624ae198067	219648	4096	6.497018	219524	1	1
4	c9b3700a77facf29172f32df6bc77f48	262144	4096	6.638142	261943	1	1

5 rows × 2030 columns

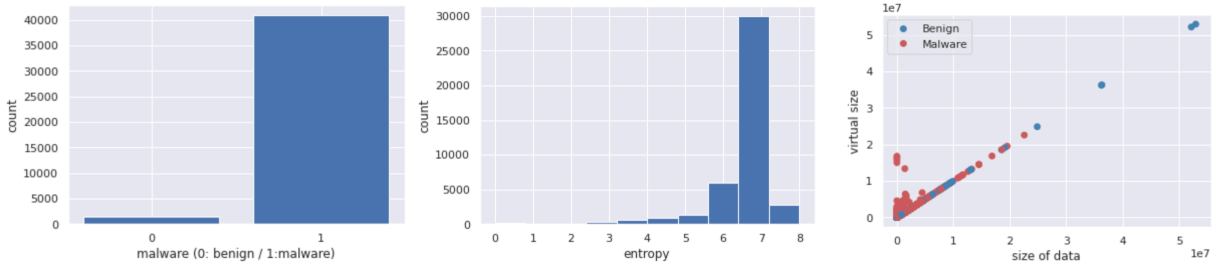
2) 將重複出現的樣本刪除，且在清理資料時發現，重複樣本幾乎都為被錯誤標籤的樣本，因此最後選擇直接將重複出現的樣本全部移除，而不是保留其中之一。

3) 進行特徵工程，添加一個新欄位 "disk_memory_ratio"，此欄位是從原有欄位 size_of_data / virtual_size 計算而得，此欄位表達的是該程式的程式碼內容大小在"儲存於硬碟時"與"儲存於記憶體時"的比值。

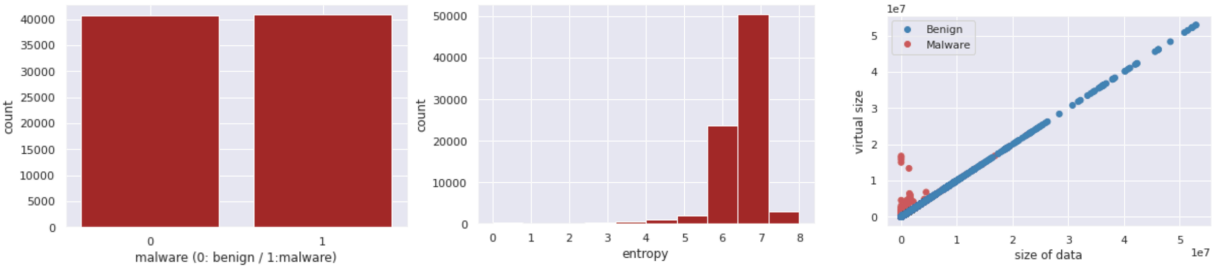
	hash	size_of_data	virtual_address	entropy	virtual_size	disk_memory_ratio
0	071e8c3f8922e186e57548cd4c703a5d	443392	4096	6.529624	442984	1.000921
1	33f8e6d08a6aae939f25a8e0d63dd523	331264	4096	6.604314	330784	1.001451
2	b68abd064e975e1c6d5f25e748663076	74240	4096	6.046789	73819	1.005703
3	72049be7bd30ea61297ea624ae198067	219648	4096	6.497018	219524	1.000565
4	c9b3700a77facf29172f32df6bc77f48	262144	4096	6.638142	261943	1.000767

5 rows × 2031 columns

4) 進行探索型資料分析 (EDA)，首先查看 Malware 與 Benign 的數量分佈，以及 entropy 與 size_of_data 的分佈。這時發現 Malware 與 Benign 的數量分布差距過大，Benign 的數量遠少於 Malware，因此需要做 Resampling。



5) 採用 ADASYN 進行 Over Sampling，並查看 Over Sampling 後的結果，得知 Malware 與 Benign 的數量分佈趨近相同，且 entropy 的分佈與原先分佈很相似。在 size_of_data 的分佈上，因為 ADASYN 是採取在同類的兩點中間加入新的一個點，並添加微小噪聲的概念，因此 ADASYN resampling 後的 size_of_data 分佈會產生明顯變化，在對角線部分，因為在原先兩點中間加入新的一個點，因此對角線的直線分佈會變得非常明顯。

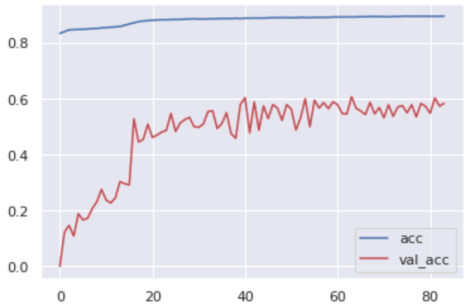


六、個別模型建立與訓練

- First Model: PE Section Headers

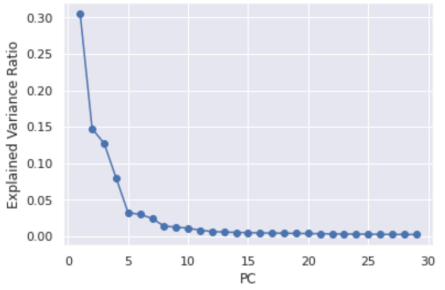
- Standardization
- Build DNN model
  - Dense(32) + Dense(32) + Dense(64) + Dropout(0.2) + Dense(1)
  - Use Adam optimizer with learning rate = 0.0003 and Early stopping
- Result : Training Accuracy: 89.71% , Validation Accuracy: 58.46%

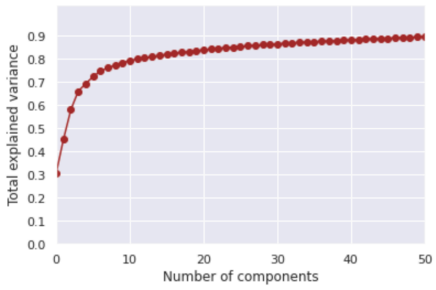
```
The training accuracy of section header model: 89.71%
The validation accuracy of section header model: 58.46%
```



- Second Model: Top-1000 PE Imports

- 因為此資料集維度偏高，因此先透過 PCA 降維，首先查看欲選取的主成份(PC)數量
- 決定選擇 60 個主成份 (PC)，共解釋 90.4% 的總變異 (Total explained variance)

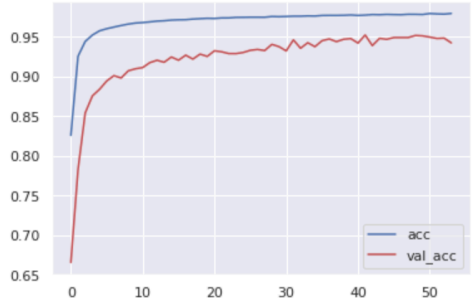





```
選擇 10 個主成份，可以解釋的總變異為 78.42 %
選擇 20 個主成份，可以解釋的總變異為 83.53 %
選擇 30 個主成份，可以解釋的總變異為 86.27 %
選擇 40 個主成份，可以解釋的總變異為 88.07 %
選擇 50 個主成份，可以解釋的總變異為 89.38 %
選擇 60 個主成份，可以解釋的總變異為 90.4 %
選擇 70 個主成份，可以解釋的總變異為 91.24 %
選擇 80 個主成份，可以解釋的總變異為 91.95 %
選擇 90 個主成份，可以解釋的總變異為 92.56 %
```

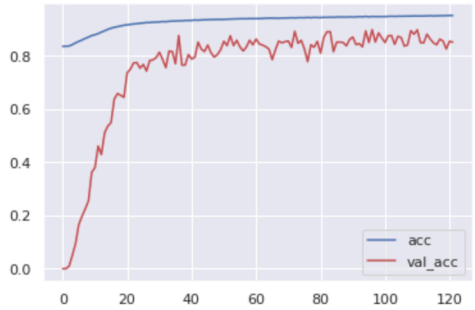
- Build DNN model
  - Dense(64) + Dense(64) + Dropout(0.4) + Dense(32) + Dense(32) + Dropout(0.2) + Dense(1)
  - Use Adam optimizer with learning rate = 0.0001 and Early stopping
- Result : Training Accuracy: 97.97% , Validation Accuracy: 94.26%

```
The training accuracy of section header model: 97.97%
The validation accuracy of section header model: 94.26%
```



- **Third Model: Raw PE as Image**
  - Min-Max Normalization (From [0, 255] to [0, 1])
  - Reshape to (32, 32, 1)
  - Build CNN model
    - Input + Conv2D(32, 4*4) + Conv2D(64, 4*4) + MaxPooling2D(2*2) + Conv2D(128, 4*4) + Conv2D(128, 4*4) + MaxPooling2D(2*2) + Flatten + Dense(256) + Dropout(0.4) + Dense(1)
    - Use Adam optimizer with learning rate = 0.000003 and Early stopping
  - Result : Training Accuracy: 95.11% , Validation Accuracy: 85.1%

```
The training accuracy of section header model: 95.11%
The validation accuracy of section header model: 85.1%
```



七、 模型集成 (Model Ensemble)

1) Build and Train the Ensemble Model: 結合上面三個 Model，並在這三個 Model 的 Output 後面再加上一層 Dense Layer with 16 neuron，最後再接上 Dense Layer with 1 neuron 做為最終 Ensemble model 的 output。

```
[ ] def model_ensemble(models, model_input):
        outputs = [model.outputs[0] for model in models]
        y = keras.layers.Dense(8, activation="relu")(outputs)
        y = keras.layers.Dense(1, activation="sigmoid")(y)
        model = Model(model_input, y, name="model_ensemble")
        return model
```

```
[ ] # Load models
    header_dnn = header_model()
    import_dnn = import_model()
    image_cnn = image_model()
```

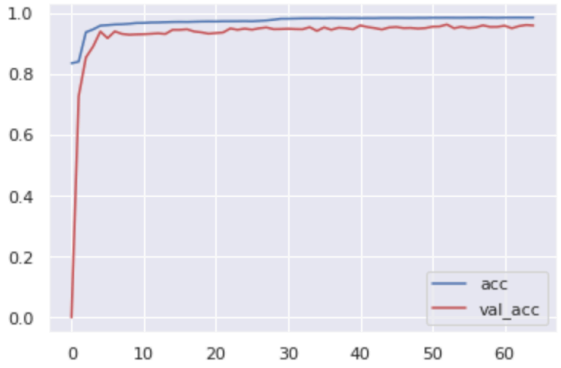
```
[ ] # combine the outputs of these three models
    combinedInput = keras.layers.concatenate([header_dnn.output, import_dnn.output, image_cnn.output])
    # Then add a Dense Layer in the end
    x = keras.layers.Dense(16, activation="relu")(combinedInput)
    x = keras.layers.Dense(1, activation="sigmoid")(x)
    # The final ensemble model
    model = keras.models.Model(inputs=[header_dnn.input, import_dnn.input, image_cnn.input], outputs=x)
```

```
[ ] model.compile(optimizer=keras.optimizers.Adam(learning_rate=3e-4),
                  loss=keras.losses.BinaryCrossentropy(),
                  metrics=['accuracy'])
    file_path = "model_ensemble.hdf5"
    early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=12)
    checkpoint = keras.callbacks.ModelCheckpoint(file_path, monitor='loss', verbose=0, save_weights_only=True, save_best_onl
    history = model.fit([header_X, import_X, image_X], y, epochs=300, callbacks=[early_stopping, checkpoint], validation_spl
    model.save(file_path)
```

2) The Result of Ensemble Model:

- Use Adam optimizer with learning rate = 0.0003 and Early stopping
- Result : Training Accuracy: 98.58% , Validation Accuracy: 95.99%

```
The training accuracy of section header model: 98.58%
The validation accuracy of section header model: 95.99%
```



八、個別模型與集成模型比較

	Training Accuracy	Validation Accuracy
PE Section Headers with DNN	**89.71%**	**58.46%**
Top-1000 PE Imports with DNN	**97.97%**	**94.26%**
Raw PE as Image with CNN	**95.11%**	**85.1%**
Ensemble Model	**98.58%**	**95.99%**

九、結論

由模型訓練結果可以看出，Model Ensemble 確實對於 Malware Detection 準確率的提升是有幫助的，且相較於個別的模型，Model Ensemble 的準確率不論是在 Training Data 或是 Validation Data 上都是最高的。

此外，PE Section Headers 的 DNN Model 在 Validation Data 上的準確率特別低，推斷可能的原因是因為該個別資料集的欄位數量過少，僅有 6 個欄位，因此單獨訓練該資料集得到的結果確實會不盡理想。

最後，對於 Malware Detection，有很多分析方式可以達成，並且有很好的準確率。然而，對於某一支程式，若是我們可以分析並提取出他不同面向的特徵，無論是 Header data 或是 Import Library，甚至是轉換成影像的程式碼內容，則透過先將這些特徵各自用最適合的模型訓練，再將其結果做合併，所得到新的結果，透過此報告的 Model Ensemble 模型表現可以得知，確實會有更好的準確率。此外，若是只將不同模型的結果以多數決或平均的方式 Ensemble 來得到新的結果，則會忽略"不同模型與特徵會對結果預測有不同影響力"這件事，因此若是改成在許多模型的預測結果之後，加入一層 Dense Layer，讓 Neural Network 自己去學習出各個模型對於最終結果的預測上，應該有的最適當的權重，如此在 Model Ensemble 的建立上會是較佳的做法。