中華民國第 58 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

第三名

團隊合作獎

052512

人工智慧化病毒掃描系統開發與探討

學校名稱：國立中央大學附屬中壢高級中學

作者：	指導老師：
高二 劉立行 高二 葉季儒	黃憲銘

關鍵詞：Supervised learning、Malware Detection、artificial neural network

摘要

一個優良的防毒軟體應該要盡可能的減少病毒對電腦造成的損害，然而面對千變萬化的網路世界，必要先發制人的預測病毒，本研究基於機器學習開發一套智慧型病毒掃描系統，運用人工神經網路訓練模型，進而由大數據進行預測。

本專案分為兩個階段，第一層是基於檔案中的各種特徵做靜態分析，第二層是基於執行檔做動態行為分析，且兩者皆採雲端化架構。

雲端掃毒雖非創舉，但本系統有著極高的病毒偵測能力，且無大量資料庫做後台比對，再複雜的變種病毒都難逃本程式法眼，以前陣子鬧得沸沸揚揚的 Wanna Cry 勒索病毒為例，我們獨自研發的人工智慧病毒掃描系統在沒有看過其原始碼的前提下，就能將其揪出並加以制止。

壹、研究動機

生活在這資訊發達的時代，我們是如此的幸福，但是面對許多未知的威脅，我們又該怎麼辦呢?本研究正是以此為原動力，想辦法盡可能降低對人類的傷害，經過反覆的思考與討論，我們想利用生活中既有的經驗作為資料，找到危險發生前的徵兆，舉例來說，程序員在觀察病毒時，會從各個不正常的操作中分析，只要發現有異常的狀況，很容易就能發現有病毒在作祟。另一個例子是犯罪，犯人通常不會突

然犯罪，一定會有所準備，例如有危險性言論，購買危險器械，等等犯罪共同特徵。所以藉由人工智慧（Artificial Intelligence, AI）， 就像一個有經驗的警察，隨時盯著犯罪的前兆，就能快速地抓出問題。

貳、研究目的

綜合上述討論，我們將設計出一套能夠學會病毒特徵的人工智慧模型，並實際開發出基於此的病毒掃描系統，此系統能藉由自行學會的病毒徵兆來判斷有害程式。

藉由本研究，我們期望驗證人工智慧的可行性，並且在真實環境應用機器學習，由大數據學習進行預測，希望能將它帶進人類生活，在未來能推廣到更深更全面的層次。

參、研究設備及器材

一、msi 筆記型電腦

規格： CPU　Intel® Core™ i7-3610QM 3.30GHz

RAM　16GB

作業系統　Windows10 專業版 x64

用途　機器學習的模型設計與訓練

二、ASUS 筆記型電腦

規格: CPU Intel® Core™ i5-3230M 2.60GHz

RAM 8GB

作業系統 Windows10 專業版 x64

用途 病毒相關分析、架構開發

三、趨勢科技虛擬機(兩台 VM)

規格： CPU intel™ e5 *2

RAM 4 GB

作業系統 Windows10 專業版 x64

用途 大量病毒資源分析

*感謝趨勢科技(Trend Micro™)提供相關實驗室環境與病毒樣本

四、本校資訊教室(6 台電腦)

規格: CPU intel™ i5

RAM 4 GB

作業系統 Windows7 專業版 x64

用途 架構實驗、病毒軟體測試

*感謝本校提供相關硬體資源

五、(一)語言:Python3.5.2

環境:Anaconda IDE:jupyter notebook

Deep learning 套件：Keras2.0.4

高維度矩陣運算工具：NumPy1.13.3

Python 科學運算套件：scikit-learn0.19.1

Python 數據分析包：pandas0.22.0

PEFile 套件 ： PEFile0.16

(二)語言:Java1.8.0　　IDE: Eclipse

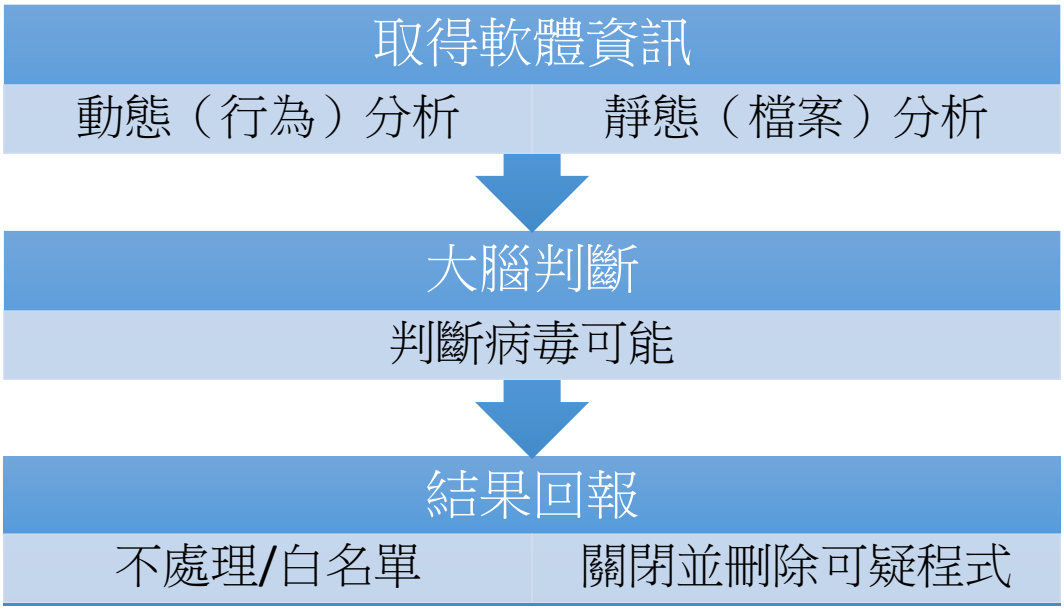
(三)語言:C#　　IDE: Visual Studio

肆、研究過程或方法

一、研究流程(圖一之一)

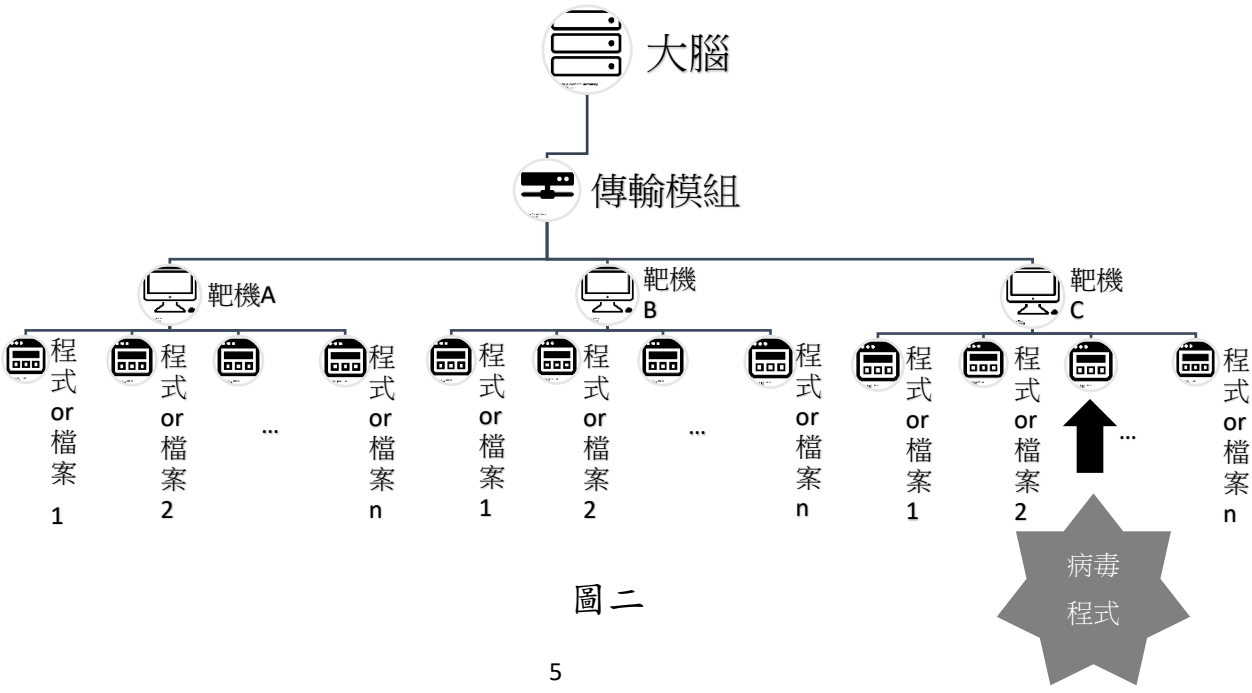
動態歷程　　靜態分析

我們採用層層的保護機制，進行病毒運行的行為分析以及 FILE 檔之各層面數組特徵，通過觀察幾項特徵判斷該軟體為病毒的機率。



二、系統架構

以下我們稱伺服端執行的神經網路為「大腦」，裝載有「病毒程式」的客戶端主機為「靶機」，以及連結兩者的「傳輸模組」。（如圖二）



圖二

1. 詳細運作過程如下(如圖三):

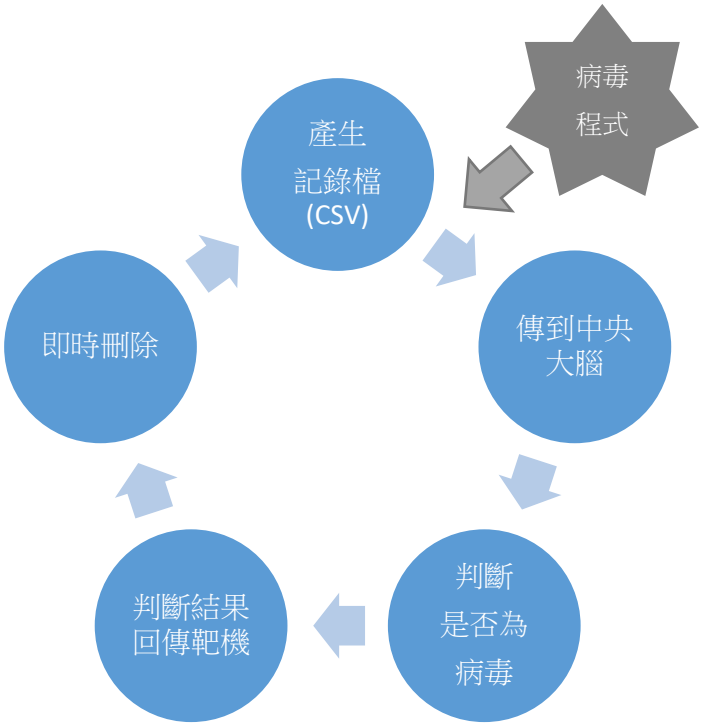
(一)傳輸模組將靶機上每個處理程序的歷程紀錄和檔案的重要特徵都集成.csv 檔，並依時間序傳輸給大腦。

(二)將加上標籤(已知結果)的資料進行前處理與特徵標準化。

(三)處理後的數據輸入至大腦中的神經網路，訓練模型。

(四)輸入大腦即時傳輸來的資料歷程紀錄作為特徵。

(五)把訓練好的模型拿來預測未知的歷程記錄，將有毒可能性高的程式回傳其辨識碼　給靶機執行移除。



圖三

2. 雲端傳輸架構優點

(一)減少使用者端負荷

(二)加強中央管理性能

(三)未來擴展性佳(Ex. 增加專用節點可將無計算能力之終端納入保護)

3. 雲端傳輸架構缺點

(一)即時傳輸對網路需求大

(二)限制傳輸的資料(Ex.複雜的系統存取紀錄無法傳輸)

四、病毒代碼實作與抓取系統

本實驗專案動態靜態分析，採用相關特徵(部分在實驗中無用之參數不會讀取，例如 name、md5 等等) 如下：

(一)動態：kill,name,FileName,ram,cpu,path,size,pid,
runnable_file_owner,runnable_file_hash_code,
runnable_file_last_access_time,
runnable_file_last_save_time
(登陸機碼與網路監聽仍在實驗中)

(二)靜態： "Kill","Name","md5","Computer",
"SizeOfOptionalHeader","Characteristics",
"MajorLinkerVersion","MinorLinkerVersion",
"SizeOfCode","SizeOfInitializedData",
"SizeOfUninitializedData","AddressOfEntryPoint"
"BaseOfCode","BaseOfData","ImageBase",
"SectionAlignment","FileAlignment",
"MajorOperatingSystemVersion",
"MinorOperatingSystemVersion",
"MajorImageVersion","MinorImageVersion",
"MajorSubsystemVersion","MinorSubsystemVersion"
"SizeOfImage","SizeOfHeaders","CheckSum",
"Subsystem","DllCharacteristics",
"SizeOfStackReserve","SizeOfStackCommit",
"SizeOfHeapReserve","SizeOfHeapCommit",

"LoaderFlags","NumberOfRvaAndSizes",

"Sectionssize","SectionsMeanEntropy",

"SectionsMinEntropy","SectionsMaxEntropy",

"SectionsMeanRawsize","SectionsMinRawsize",

"SectionMaxRawsize","SectionsMeanVirtualsize",

"SectionsMinVirtualsize",

"SectionMaxVirtualsize",

"ImportsDLLsize","Importssize",

"ImportsNbOrdinal","Exportsize",

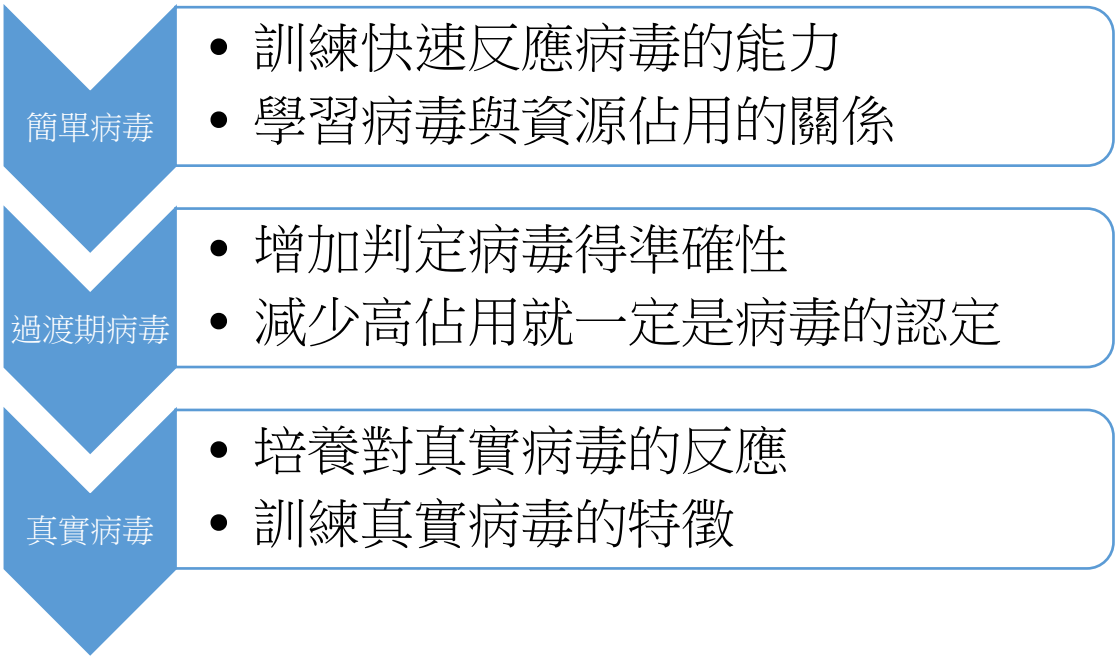
"Resourcessize",ResourcesMeanEntropy",

"ResourcesMinEntropy","ResourcesMaxEntropy",

"ResourcesMeanSize","ResourcesMinSize",

"ResourcesMaxSize","LoadConfigData",

階段性實驗(圖四)



訓練過程為了漸進式的訓練與測試大腦的判斷能力，我們一開始採用幾個特化過的病毒（每個病毒都只有一個特別顯著的特徵）訓練，詳細清單：

（1）svchost.exe ---高度 ram 佔用

（2）Firefox.exe ---高度 disk 佔用

（4）system.exe ---高度 cpu 佔用

接著再使用一些輕微特徵的病毒訓練，減少過度的獨斷、強化特徵辨識的能力:

（1）svchost.exe --- 中度 ram 佔用

（2）target.exe ---中度 cpu 佔用

（3）chrome.exe ---中度 ram 佔用

最後再使用一般的病毒測試，判斷查殺的準確度:

(1)Adobe_Update.exe ---無特別突出之特徵

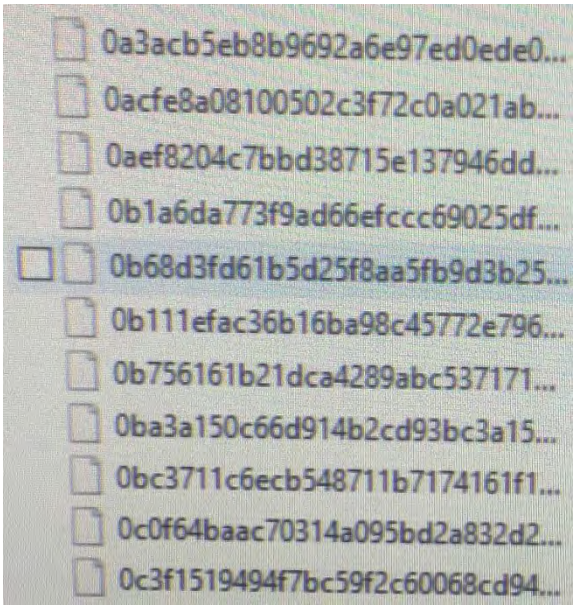
*病毒名字係用於擾亂判斷，並無特定意義

```
static void deleta()
{

    int[] sort = new int[3];
    sort = getsort();
    for(int i = 0; i < 3; i++)
    {
        Console.WriteLine("delete" + sort[i]);
        String newpath = path + "text" + sort[i] + ".txt";
        StreamWriter sw = new StreamWriter(newpath);
        sw.WriteLine("");
        sw.Flush();
        sw.Close();
    }

}
```

目標:破壞指定檔案(圖五)

(2)WannaCry 等大型病毒

(3)測試趨勢科技資料庫其他大量病毒(圖六)



五、掃毒引擎開發

(一)import 函式庫與.csv 格式(逗號分隔值)資料

(二)資料前處理函式包裝為 preprocessData(如圖七)

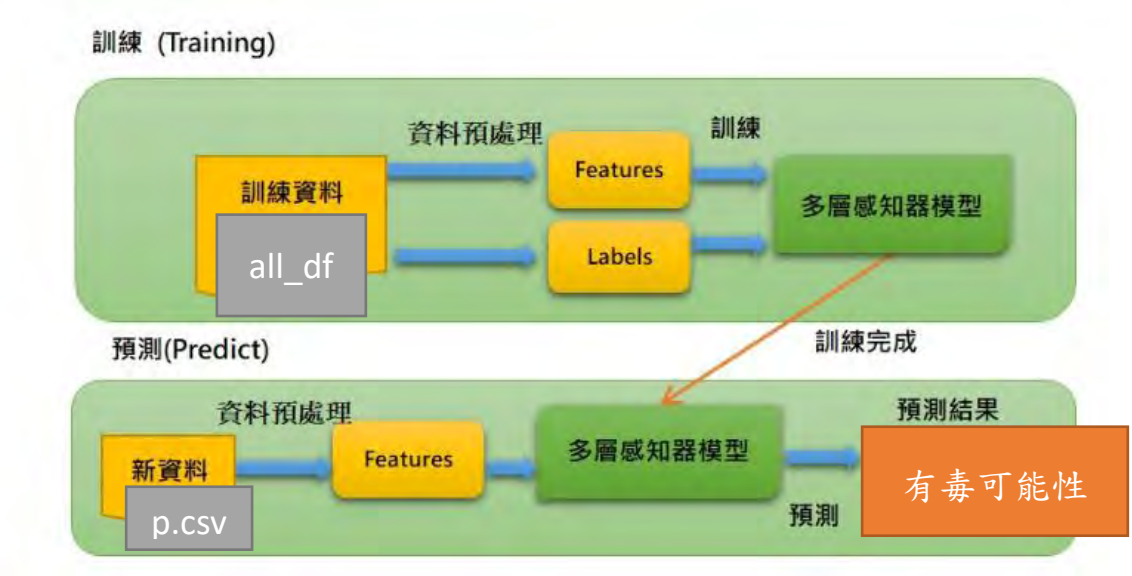
1. 將 dataframe 轉換為 array

2. 將 ndarray 特徵欄位進行標準化

3. 將 path 等字符特徵欄位使用 one hot vector 編碼

```
29 def preprocessData(raw_df):
30
31     df = raw_df
32     df=raw_df.drop(['name'], axis=1)
33     #size_mean = df['size'].mean()
34     #df['size'] = df['size'].fillna(size_mean)
35     x_OneHot_df = pd.get_dummies(data=df,columns=["path"])
36
37     ndarray = x_OneHot_df.values
38     Label = ndarray[:,0]
39     Features = ndarray[:,1:7]
40
41     minmax_scale = preprocessing.MinMaxScaler(feature_range=(0, 1))
42     scaledFeatures=minmax_scale.fit_transform(Features)
43
44     return scaledFeatures,Label
```

圖七



圖八

(三) Create Model (如圖九)

本專案實驗過 logit regression 與 k-nearest neighbors 效果均不是很明顯，另外 decision tree 尚未實作完成，所以沒有納入演算法的選擇與評比。

最後決定使用 NN(人工神經網路)

1. 導入 keras 模組

2. 建立序列式模型

3.建立神經層(如圖十）

輸入層(import layer)有 1024 個神經元

隱藏層(hidden layer)有 512 個神經元

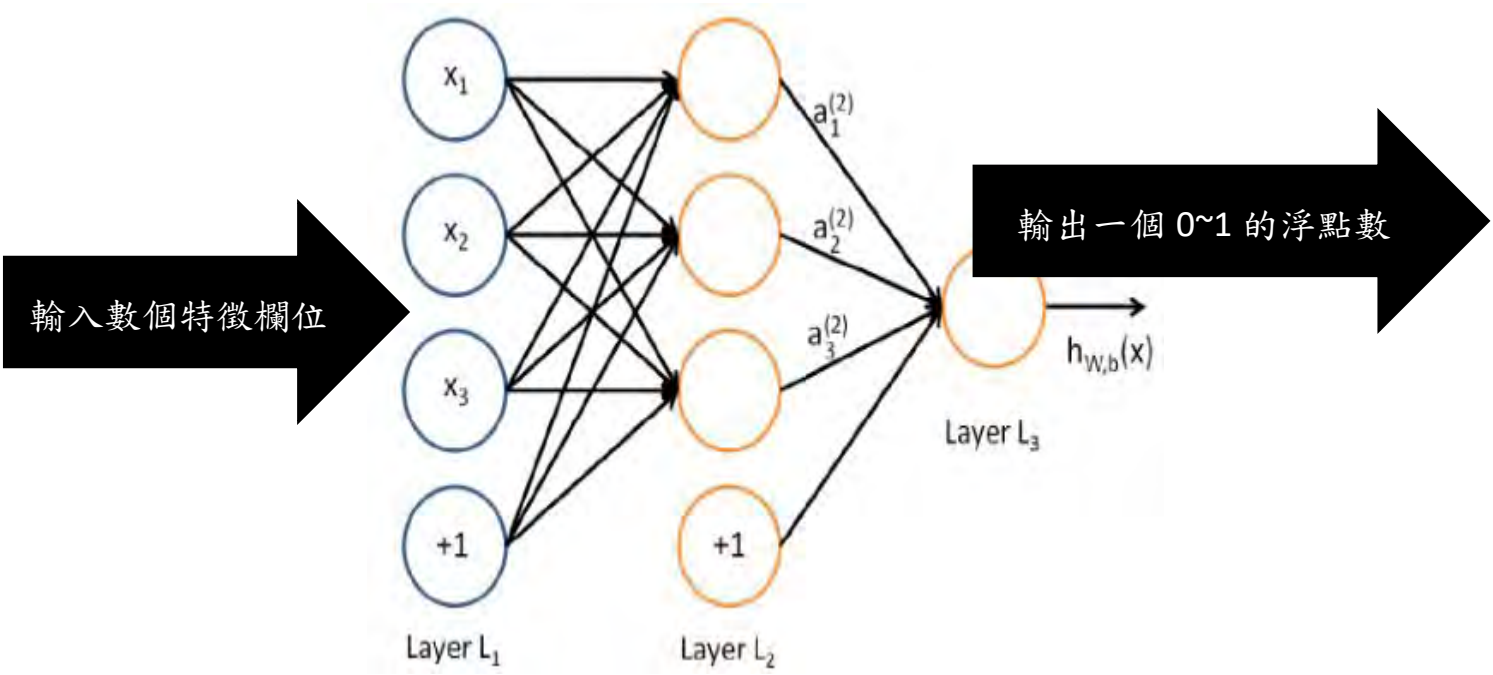
前兩層接採用線性整流函數(Rectified Linear Unit, ReLU)作為激勵函數，激勵函數的主要作用是引入非線性，如果還是矩陣的線性運算則使神經網路毫無意義。

輸出層有一個經過 sigmoid 壓縮的小數輸出代表有毒可能性（機率: 愈接近 0 表示愈不可能是病毒，愈接近 1 則愈有可能是病毒）

4.藉由 Backpropagation 演算法反覆計算梯度優化權重，目標函式降低模型 loss。

```
model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['acc'])

train_history =model.fit(x=train_Features,
                         y=train_Label,
                         validation_split=0.1,
                         epochs=100,
                         batch_size=20,verbose=2)
```

圖九

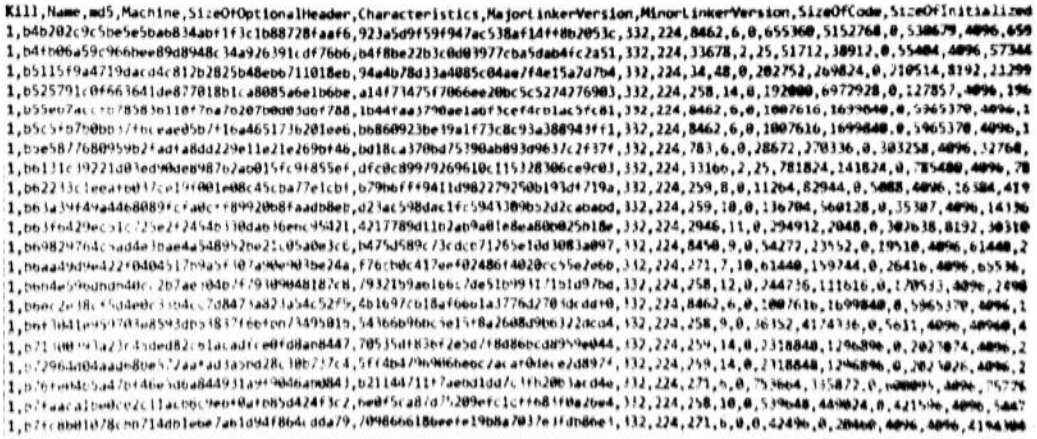


圖十

（五）實驗環境統整（圖十一）

伍、研究結果

擷取部分 csv 檔內容(如圖十二)：




```
kill,name,ram,cpu,path,size,pid
0,svchost,16519168,2.65625,System32\,37256,1772
0,jhi_service,987136,0.015625,DAL\,176416,5908
0,winlogon,2588672,3.953125,system32\,0,584
0,IntelCpHDCPSvc,655360,0.078125,system32\,0,1764
0,TeamViewer_Service,13197312,1.375,TeamViewer\,10945776,1960
0,java,21778432,1.90625,javapath\,191040,3928
0,ApplicationFrameHost,21475328,2.640625,system32\,0,5148
0,LogMeInSystray,6541312,0.578125,x64\,423424,5696
0,LMIGuardianSvc,3952640,0.234375,x64\,419304,1952
0,jusched,1163264,0.046875,Java Update\,587288,6084
0,Windows10UpgraderApp,2260992,2.546875,Windows10Upgrade\,612528,4304
0,hamachi-2-ui,13955072,3.515625,LogMeIn Hamachi\,5885352,5876
0,svchost,34598912,36.0625,system32\,37256,948
0,igfxCUIService,806912,0.09375,system32\,0,1144
0,sihost,14168064,3.703125,system32\,0,3308
0,svchost,14204928,5.015625,system32\,37256,1332
0,svchost,11751424,5.328125,system32\,37256,740
0,svchost,13348864,1.984375,System32\,37256,4992
0,conhost,13811712,0.140625,system32\,0,1720
0,csrss,1507328,2.34375,,0,428
0,explorer,117940224,13.5,Windows\,4502864,3688
0,smss,139264,0.5,,0,324
0,dwm,46161920,27.20313,system32\,0,908
0,svchost,9596928,15.5625,system32\,37256,1300
0,svchost,11214848,4.234375,System32\,37256,508
0,csrss,22499328,17.28125,,0,504
0,LogMeIn,13115392,2.234375,x64\,407424,1284
0,MsMpEng,117174272,59.59375,,0,1840
0,RuntimeBroker,30363648,27.98438,System32\,0,3448
0,SystemSettings,61440,0.75,ImmersiveControlPanel\,83704,5216
```

*kill是訓練過程中判斷是法有毒的標籤，在正式測試中會被移除

圖十二

擷取部分 pandas 數據框架的 dataframe(如圖十三)：

```
In [14]: pd[:70]
```

Out[14]:

	kill	name	ram	cpu	path	size	pid	probability
0	0	conhost	17444864	0.171875	system32\	0	824	0.048297
1	0	Taskmgr	26820608	26.765630	system32\	1083136	5504	0.344183
2	1	svchost	433827840	3.718750	process\	5632	3472	0.758618
3	1	svchost_1	433827840	3.718750	process\	5632	3472	0.758618
4	1	svchost	441827840	3.719950	process\	5632	3472	0.760831
5	1	svchost_1	435827840	3.728750	process\	5632	3472	0.759203
6	0	MpCmdRun	5914624	0.281250	Windows Defender\	370272	3632	0.093875
7	0	csrss	1818624	1.546875	NaN	0	436	0.041667
8	0	jhi_service	892928	0.031250	DAL\	176416	5900	0.007647
9	0	Test1	15581184	0.968750	process\	8192	3104	0.094347
10	0	OneDrive	4038656	1.578125	OneDrive\	1558688	2124	0.051535
11	0	wininit	1585152	0.359375	NaN	0	520	0.040647
12	0	svchost	10645504	1.671875	system32\	37256	1320	0.055604
13	0	SearchIndexer	11341824	4.687500	system32\	759808	4100	0.121156
14	0	ACDSeeCommanderPro8	3338240	0.359375	8.0\	2136584	5220	0.007220
15	0	taskhostw	4997120	1.000000	system32\	0	3524	0.098443
16	0	LogMeInSystray	4141056	0.468750	x64\	423424	4364	0.113303
17	0	csrss	7680000	4.343750	NaN	0	512	0.048069
18	0	svchost	33734656	40.687500	system32\	37256	956	0.304517
19	0	MsMpEng	76173312	20.578130	NaN	0	1844	0.194889
20	0	hamachi-2-ui	3866624	2.765625	LogMeIn Hamachi\	5885352	3840	0.042619
21	0	RuntimeBroker	12034048	11.000000	System32\	0	3352	0.137908
22	0	winlogon	2813952	0.656250	system32\	0	592	0.041929
23	0	acdIDInTouch2	5017600	0.218750	8.0\	1814800	2460	0.004296
24	0	svchost	6488064	4.140625	system32\	37256	768	0.050050

圖十三

每 10 筆採樣訓練紀錄：

Epoch 1/100

0s - loss: 0.6928 - acc: 0.5094 - val_loss: 0.6917 - val_acc: 0.8333

Epoch 10/100

- 0s - loss: 0.6760 - acc: 0.7925 - val_loss: 0.6720 - val_acc: 0.8333

Epoch 20/100

- 0s - loss: 0.6143 - acc: 0.7925 - val_loss: 0.6095 - val_acc: 0.8333

Epoch 30/100

- 0s - loss: 0.4924 - acc: 0.7925 - val_loss: 0.5163 - val_acc: 0.8333

Epoch 40/100

- 0s - loss: 0.3960 - acc: 0.7925 - val_loss: 0.4801 - val_acc: 0.8333

Epoch 50/100

- 0s - loss: 0.3021 - acc: 0.9057 - val_loss: 0.4476 - val_acc: 0.8333

Epoch 60/100

- 0s - loss: 0.2193 - acc: 0.9245 - val_loss: 0.4193 - val_acc: 0.8333

Epoch 70/100

- 0s - loss: 0.1628 - acc: 0.9811 - val_loss: 0.3865 - val_acc: 0.8333

Epoch 80/100

- 0s - loss: 0.1323 - acc: 0.9811 - val_loss: 0.3643 - val_acc: 0.8333

Epoch 90/100

- 0s - loss: 0.1177 - acc: 0.9811 - val_loss: 0.3446 - val_acc: 0.8333

Epoch 100/100

- 0s - loss: 0.1101 - acc: 0.9811 - val_loss: 0.3215 - val_acc: 0.8333

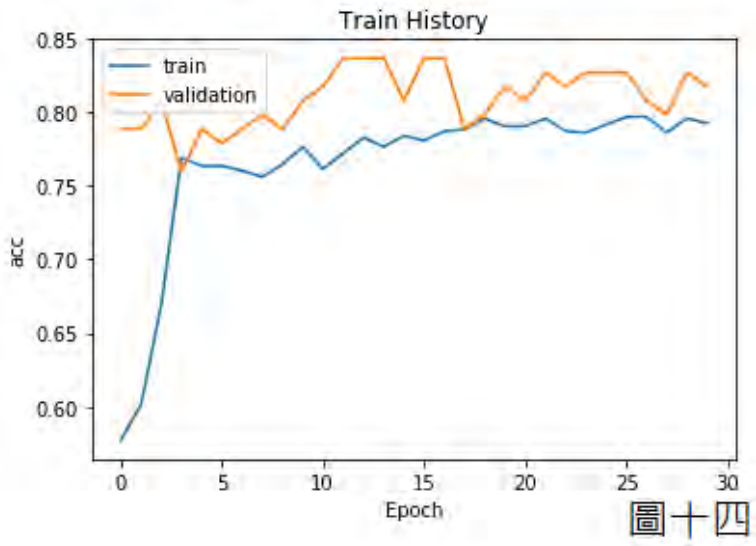
val 代表 validation，是從訓練資料中切分除來的驗證集，

如果 val_loss 很低了，可是測試集的 loss 沒有很有成果的話表示，

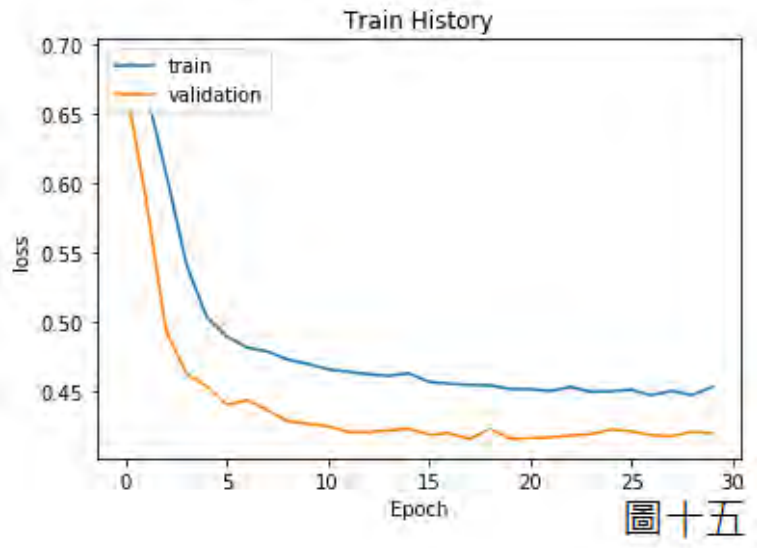
此模型對於訓練集的過度擬合(over-fitting)，則測出來的數據毫無意義，

val_loss 很低只是一種假象。

觀察我們訓練的模型，取前 30 epoches 會發現此模型並無 Over-Fitting 的狀況，訓練與測試資料皆於後期有達到預期成果。



圖十四



*epoch 就是整個資料集被訓練演算法遍歷的次數，run 完一次為 1 epoch

由上可見神經網路模型錯誤率由最初的 0.6928 降到最後 0.1101，確定此優化算法有效(圖十四、十五)。

再來切割 1/10 的資料做測試，得模型準確率約為 0.92。

```
[68 rows x 8 columns] scores: 0.928571403027
```

最後不斷傳送靶機的資料給大腦，然後開啟病毒程式，

隨機從 30130 挑選 300 種病毒經過 300 次的實驗後，有 266 次電腦成功移除病毒。(一次實驗一個病毒)

成功利用病毒徵兆發現病毒

下圖病毒掃描系統預判 PID 為 5460 的 svchost 有 0.97(97%)的危險度，Svchost 正為我們植入實驗系統中的病毒。

(圖十六)

```
pd_kill:

     kill     name        ram       cpu      path  size   pid  probability
23      0  svchost  455389184  5.515625  process\  5632  5460      0.97936
```

陸、結論

面對有害電腦系統運作的資源消耗型病毒，本研究特別能從其層層包裝中將其繩之以法，例如蠕蟲病毒，抑或是未經驗驗證的背景挖礦程式，一旦符合機器所認為的病毒特徵，本系統便能將其指出。

對於大型有害病毒有可能存取登陸檔或經由網路存取特定資源亦可有效的發現。

相較於基於病毒定義庫的舊式防毒軟體，本研究更能凸顯其預測之特色，面對未知病毒，基於人工智慧的系統表現是與時俱進的，及時把傷害降到最低，本研究驗證了機器學習實際應用的效能與可行性，雖然成功擊殺率約只達 88%(即 266/300)，卻是在完全沒有人工參與的情況下達成，若能有更高階的運算資源以及蒐集其他各種層面更完整的特徵集，大腦的訓練效果將遠不止於當前研究，勢必能順利守護資料安全。

正常程式	**共 450000 個 (+-2%)**	**誤殺**	**共 14 個 (300 次實驗)**
惡意程式	**共 30130 個**	**成功擊殺**	**共 266 個 (300 次實驗)**

目前的總實驗數據量(表一)

柒、討論

經過一段時間的訓練和測試，我們發現訓練出的大腦已具有初步分析病毒的能力，對 svchost、system 等簡單的病毒查殺的準度準確度相當的高(約可達到 99%)，但是加上其他較複雜的病毒攔截能力就比較有限(約有 88%)，我們推測可能是因為以下原因:

△訓練時間、訓練資料不足

△訓練資料太過明顯，導致對真實病毒的判斷能力有限

△偵測的資料過少，無法完全反映病毒行為

最後我們提高病毒樣本數量，優化了運算流程，正確擊殺率顯著地提高，我們也繼續優化本系統，正因為採用 ML，效能會與時俱進，使成果更加亮眼。

(一) 人工智慧化的病毒引擎最終是否能擊敗專門的安全人員？

如果可以讓學習的資料越接近真實使用者環境,並有了更好的運算設備，那麼完全擊敗將是指日可待。

(二) 本研究是否有可能誤殺其他正常程式？

是的，絕對有可能。因為本研究只有幾個特徵欄位供機器學習，如果把能參考的特徵增加，機器就更能夠了解分析這些資料間與病毒的關聯性，進而降低誤殺其他軟體的錯誤。

(三) 未來展望：

隨著硬體極限的突破，希望能有更多其他層面的應用，交通事故預測、疾病預防及時治療、預處理犯罪維護安全等等。

本研究說明了人工智慧的前途無可限量,正等著處在資訊新世代的我們去探索。

捌、參考資料及其他

(一)KERAS 官方 API
https://keras.io

(二)https://github.com/keras-team/keras/blob/master/examples/mnist_mlp.py

(三)https://morvanzhou.github.io/tutorials/machine-learning/keras/

(四)人工智慧趨勢新聞
https://technews.tw/2017/10/05/ai-machine-learning-and-deep-learning/

(五)International Journal of Network Security & Its Applications (IJNSA) Vol.7, No.6, November 2015 DOI: A_MACHINE_LEARNING_APPROACH_TO ANOMALY-BAS ED DETECTION ON ANDROID PLATFORMS Joshua Abah1, Waziri O.V2, Abdullahi M.B3, Arthur U.M4 and Adewale O.S5

(六)國立台灣大學 李宏毅教授 開放影音
林軒田教授 機器學習基石 PDF

(七)https://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E6%95%B4%E6%B5%81%E5%87%BD%E6%95%B0

(八)IT 幫幫忙 機器學習-決策樹與 k-NN 分類器 https://ithelp.ithome.com.tw/articles/10187191

(九) https://github.com/llSourcell/antivirus_demo

(十)http://www.trendmicro.tw/vinfo/tw/threat-encyclopedia/

(十一)MIT 麻省理工學院開放式課程 2006-SUMMER-AI

(十二) https://wenqingfu.me/2014/11/08/PE_file/

(十三)https://msdn.microsoft.com/zh-tw/library/system.diagnostics.process(v=vs.110).aspx

(十四) https://github.com/HackyRoot/ml-antivirus

【評語】052512

本研究提出基於機器學習之智慧型病毒掃描系統，運用人工神經網路訓練模型，並以大數據進行病毒預測。

該研究所提架構不需要使用到大量的資料庫進行比對，而是透過訓練好的模組進行病毒的預測，可節省大量的資源。透過雲端伺服器計算的功能，可減輕使用者端的運算負荷。此系統能夠透過訓練好的人工神經網路模組偵測出特定的病毒，並對該病毒進行清除。

作品海報

人工智慧化病毒掃描系統開發與探討

摘要

一個優良的防毒軟體應該要盡可能的減少病毒對電腦造成的損害，然而面對千變萬化的網路世界，必要先發制人的預測病毒，本研究基於機器學習開發一套智慧型病毒掃描系統，運用人工神經網路訓練多層感知模型，進而由大數據進行預測。

01 | 研究目的與動機

生活在這資訊發達的時代，我們是如此的幸福，但是面對許多未知的威脅，我們又該怎麼辦呢?本研究正是以此為原動力，想辦法盡可能降低對人類的傷害，我們將設計出一套能夠學會病毒特徵的人工智慧模型，並實際開發出基於此的病毒掃描系統，此系統能藉由自行認學習病毒徵兆來判斷有害程式。

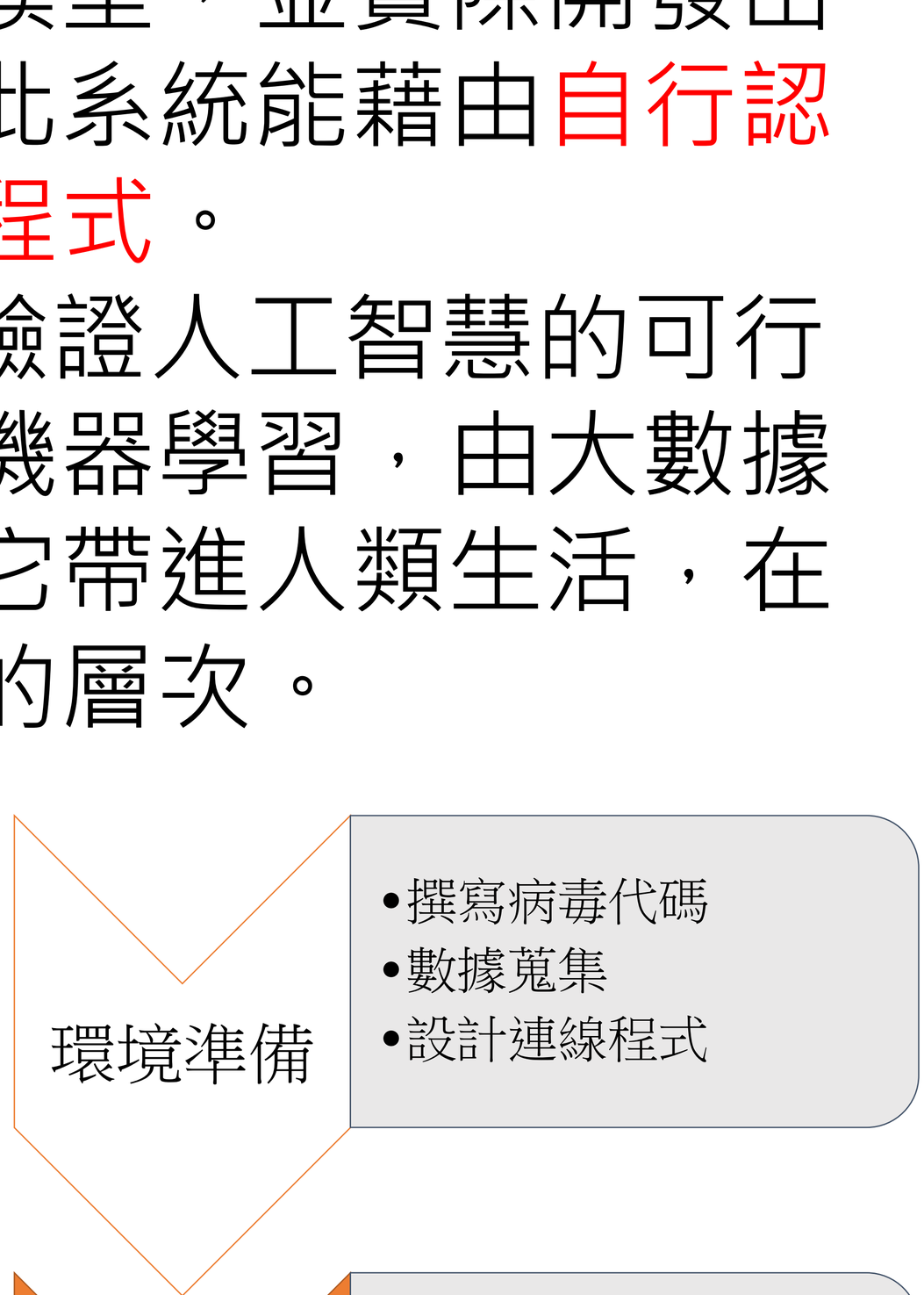
藉由本研究，我們期望驗證人工智慧的可行性，並且在真實環境應用機器學習，由大數據學習進行預測，希望能將它帶進人類生活，在未來能推廣到更深更全面的層次。

02 | 研究過程或方法

一、研究流程(如右圖)

二、系統架構

以下我們稱伺服端執行的神經網路為「大腦」，裝載有「病毒程式」的客戶端主機為「靶機」，以及連結兩者的「傳輸模組」。



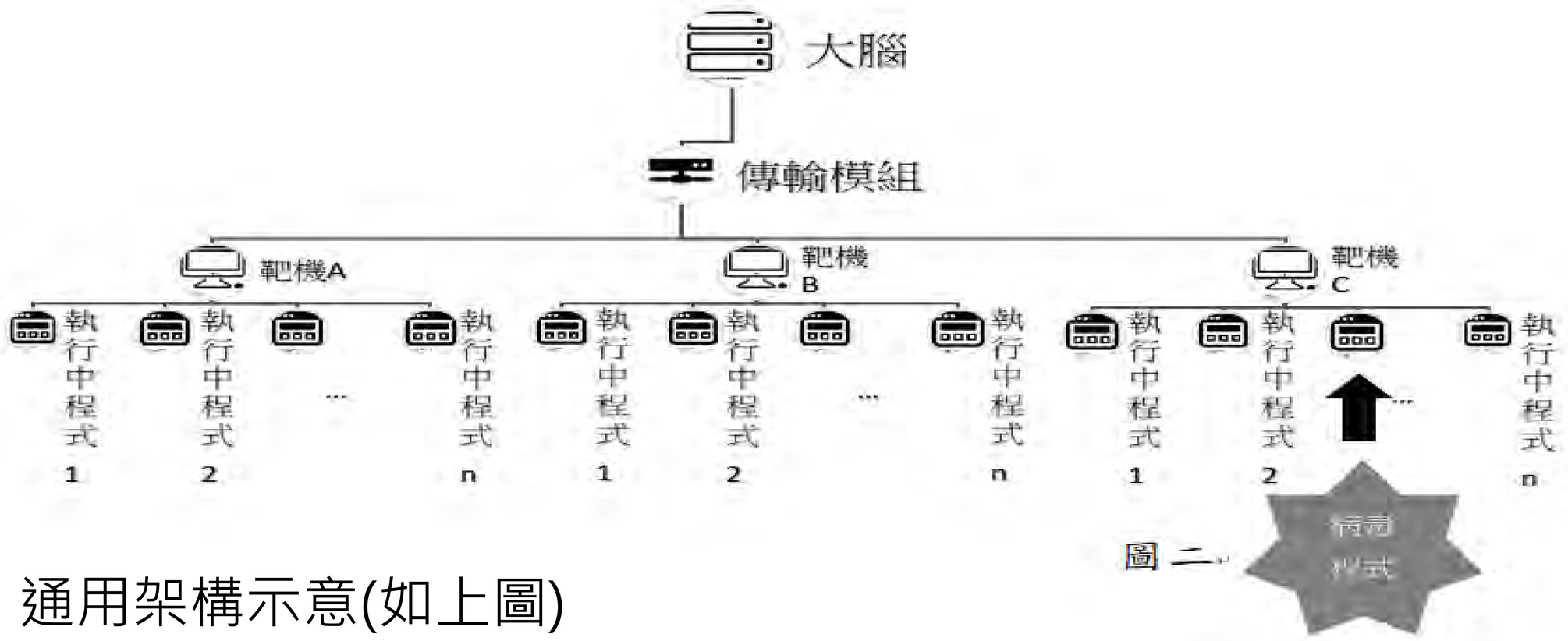


圖 二

通用架構示意(如上圖)

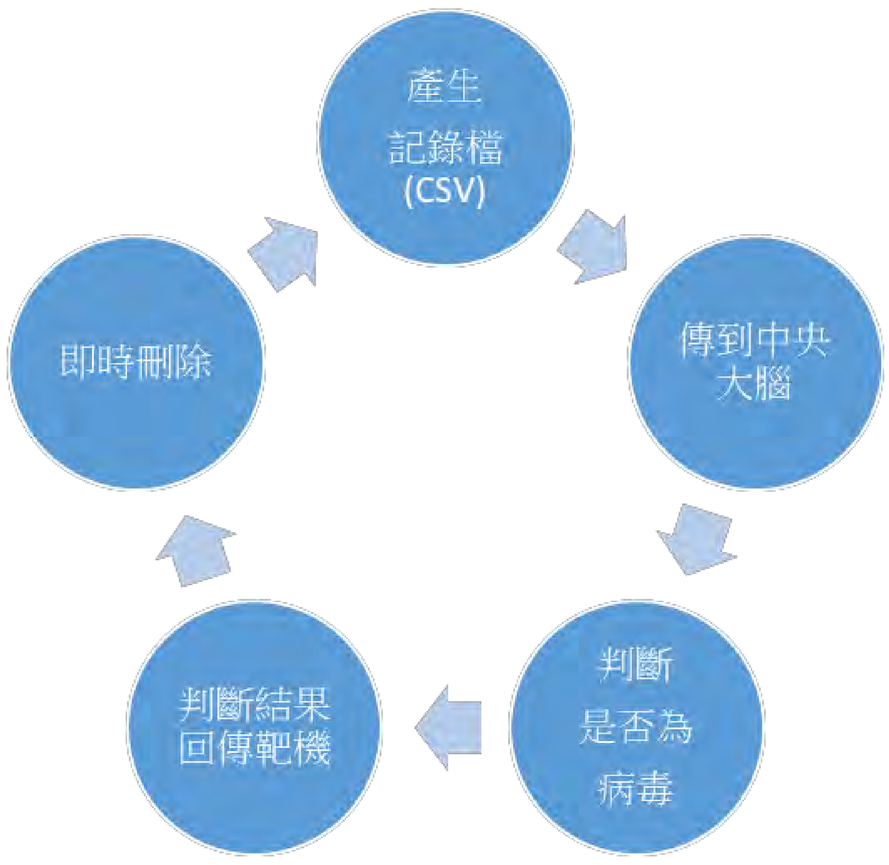
(一)傳輸模組將靶機上每個處理程序的歷程紀錄都集成.csv檔，依時間序傳輸給大腦。

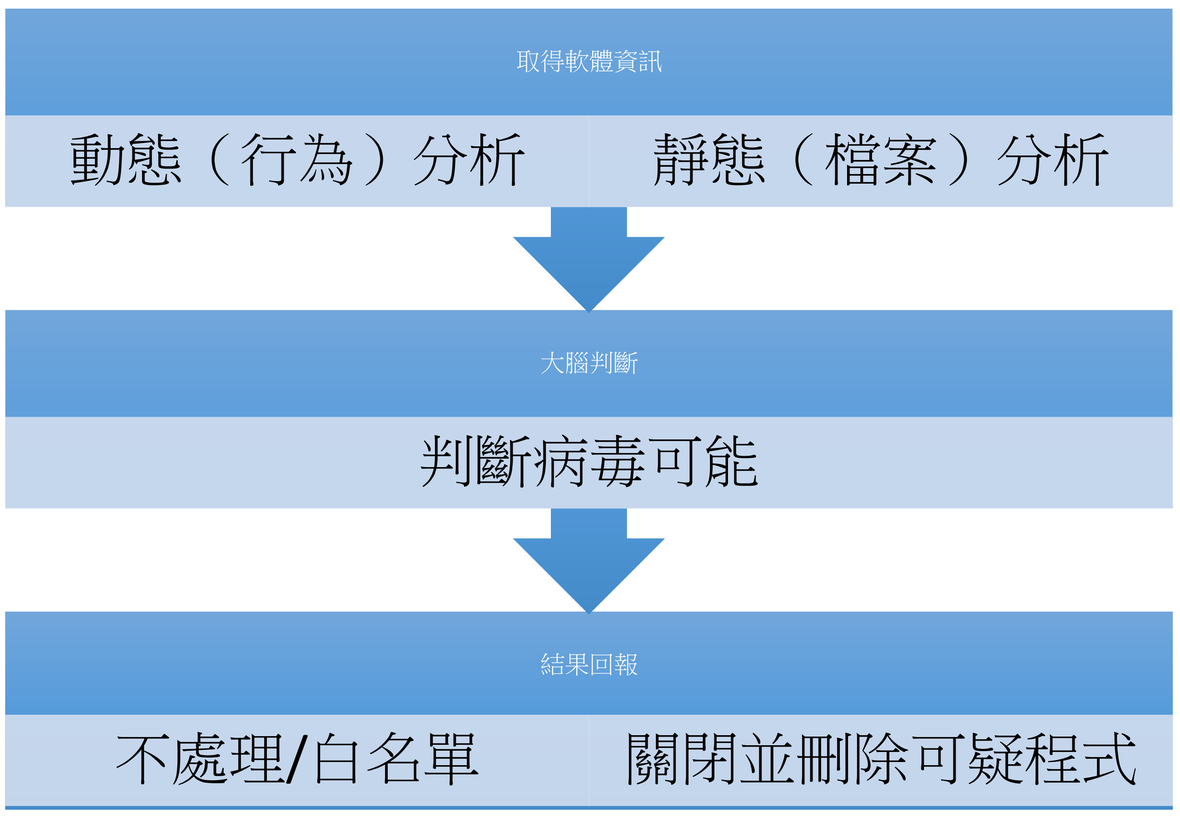
(二)將加上標籤(已知結果)的資料進行預處理與特徵標準化。

(三)處理後的數據輸入至大腦中的神經網路，訓練模型。

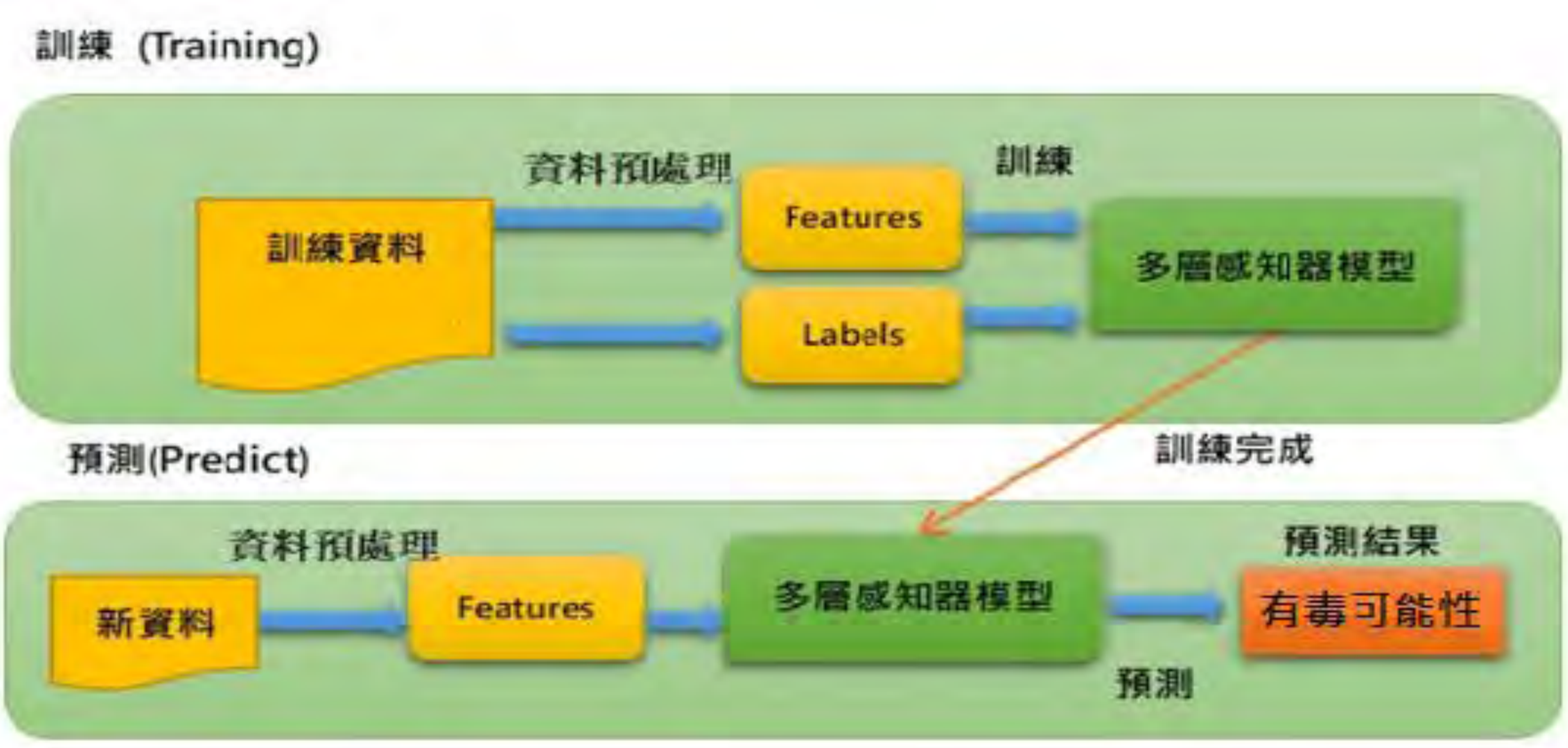
(四)輸入大腦即時傳輸來的資料歷程紀錄作為要預測的特徵。

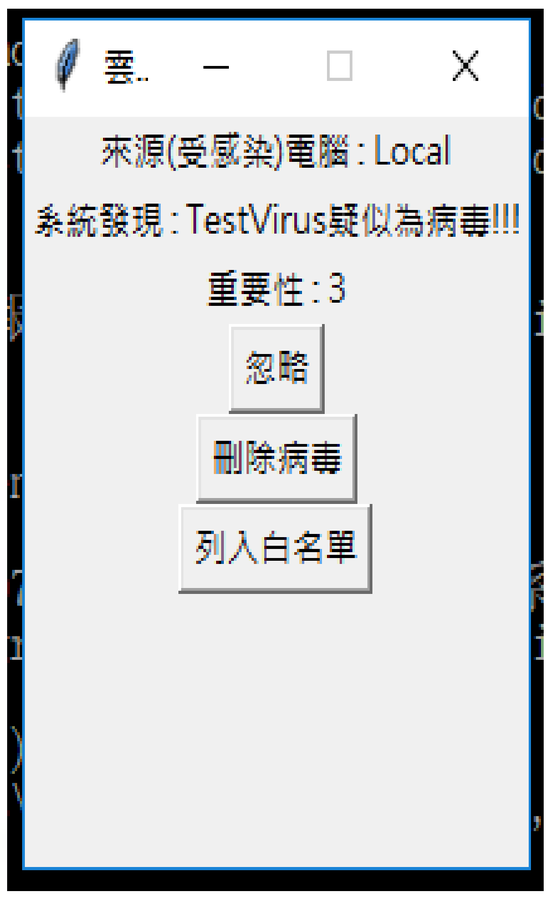
(五)把訓練好的模型拿來預測未知的歷程記錄，將有毒可能性高的程式回傳其辨識碼(pid)，給靶機執行移除。

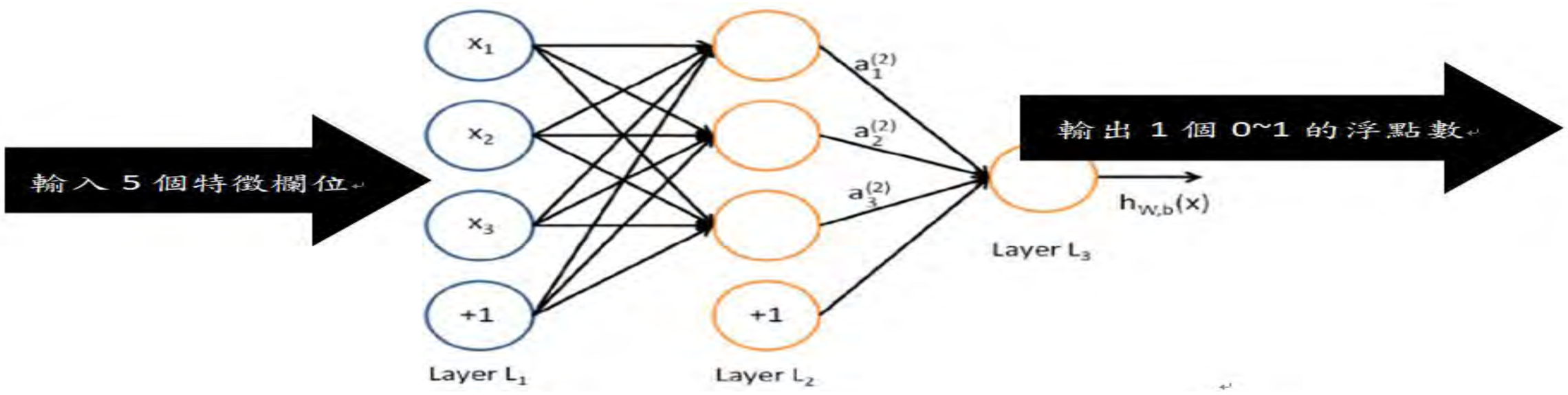




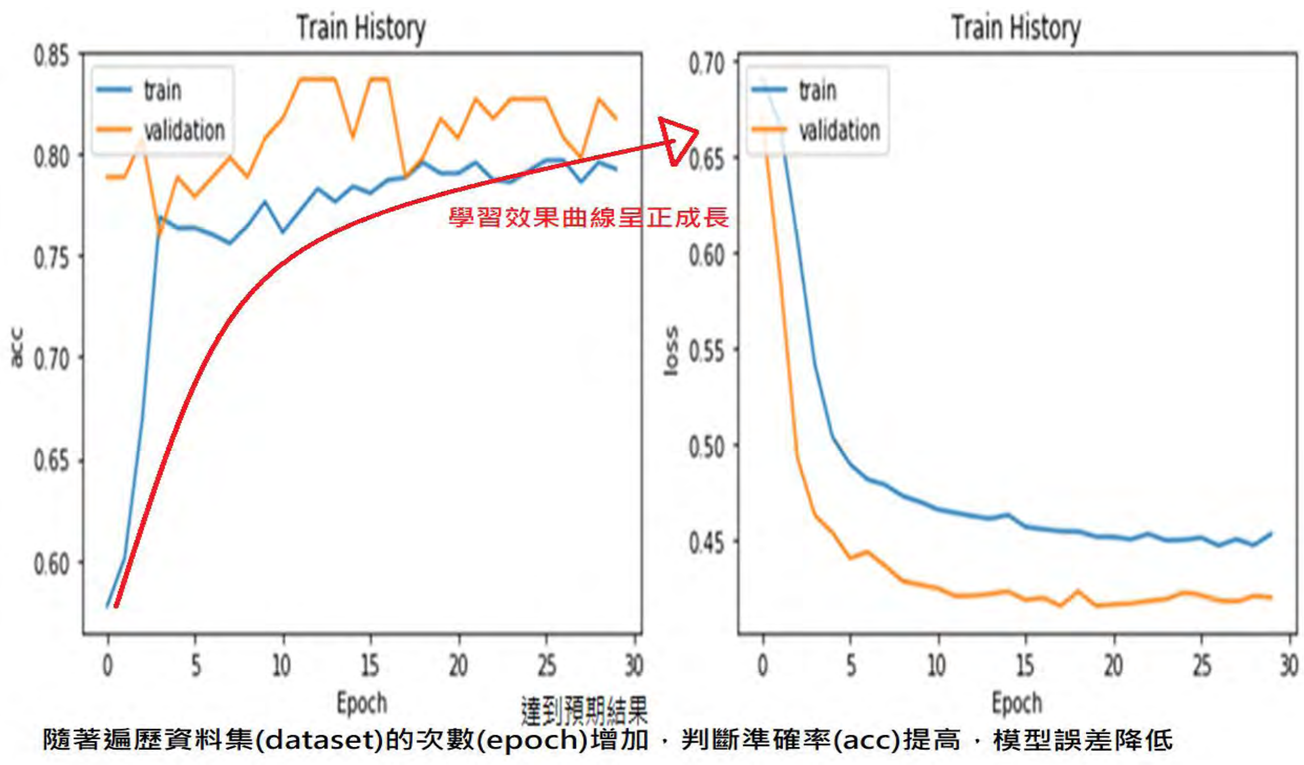
03 | 掃毒引擎開發







05 | 研究結果



隨著遍歷資料集(dataset)的次數(epoch)增加，判斷準確率(acc)提高，模型誤差降低

```
[Info] The prediction of top10 records:
   kill             name       ram       cpu   size   pid  \
0     0           svchost  11411456  0.125000  44520  3016
1     0           svchost   7286784  0.015625  44520  2152
2     0  SearchProtocolHost  14114816  0.046875  324608  6892
3     0        MasterHelper   8867840  0.062500  77824  4736
4     0           igfxTray   8433664  0.109375      0  6028
5     0     @WanaDecryptor@  10964992  0.140625  245760  8612
6     0           svchost  96698368  42.062500  44520  2112
7     0           svchost   3801088  0.015625  44520  3068
8     0           svchost   5718016  0.015625  44520  3004
9     0           winlogon   9424896  0.125000      0   848

   runnable_file_last_access_time  runnable_file_last_save_time  probability
0                  20170929214214                20170929214214     0.000493
1                  20170929214214                20170929214214     0.000444
2                  20180424100850                20180416040545     0.000595
3                  20180306094831                20161104084718     0.000535
4
5                  20180712200416                20170512022256     0.000687
6
7                  20170929214214                20170929214214     0.000484
8                  20170929214214                20170929214214     0.000464
```

```
pd_kill:

    kill     name        ram       cpu      path  size   pid  probability
23     0  svchost  455389184  5.515625  process\  5632  5460      0.97936
```

△成功舉報異常程式△

	Predicted Positive	Predicted Negative
Actual Positive	657(a)	56(b)
Actual Negative	13(c)	63(d)

Accuracy=(a+d)/(a+b+c+d)=91.2%
Precision =(a)/(a+c)=91.2%
Recall=(a)/(a+b)=91.2%
F1-measure=(2*P*R)/(P+R)

	本系統	AVG	Avast	McAfee	360衛士	賽門鐵克	Baidu	WDefender	T廠牌
eicar	△	○	○	○	○	○	○	○	○
Malware1	○	○	○	△	△	○	△	○	○
…	7	5	4	3	3	5	2	5	6
Malware11	○	○	△	○	△	○	△	△	△

△面對低識別率的惡意軟體一樣有高擊殺率

06 | 結論

本研究驗證了機器學習的效能與可行性，雖然成功擊殺模型F1-measure約只達94.9%，卻是在完全沒有人工參與的情況下達成，如果能有更進一步的研究開發，勢必能盡力守護資料安全。

07 | 未來展望

隨著硬體極限的突破，希望能有更多其他層面的應用，疾病預測及時治療、預防犯罪維護安全等等。

本研究說明了人工智慧的前途無可限量，正等著處在資訊新世代的我們去探索。

國立中央大學附屬中壢高級中學
作者 劉立行 葉季儒 指導教師 黃憲銘