

ภาคผนวก

# ขั้นตอนการติดตั้งไลบรารีและเครื่องมือสำหรับการใช้งานโครงข่ายประสาทเชิงลึกด้วยคอมพิวเตอร์ส่วนบุคคล

## ส่วนที่ 1 ส่วนประกอบที่จำเป็นในการติดตั้งโปรแกรม

### 1.1. ส่วนประกอบที่จำเป็นในการติดตั้งโปรแกรม

1.1.1. Windows 10 x64 bits

1.1.2. Python 3.7

1.1.3. Anaconda Navigator

## ส่วนที่ 2 ขั้นตอนการใช้งานและการทำงานของโปรแกรมที่เกี่ยวข้อง

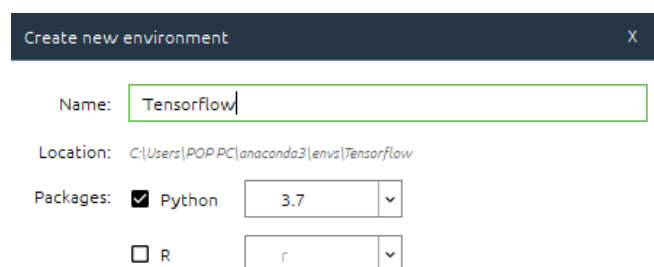
### 2.1. การติดตั้งสภาพแวดล้อมที่จำเป็นโดยใช้ Anaconda Navigator

#### 2.1.1. เข้าเว็บไซต์ และเลือกดาวน์โหลดแอปพลิเคชันสำหรับ Windows 64 bit



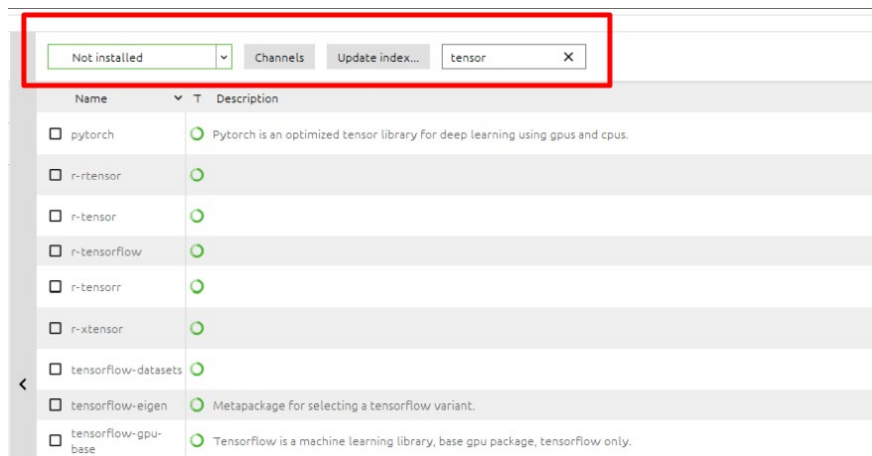
รูปที่ ผ.1 การโหลดแอปพลิเคชัน Anaconda Navigator ผ่านเว็บไซต์

#### 2.1.2. สร้างสภาพแวดล้อมใหม่เลือกเป็น Python เวอร์ชัน 3.7

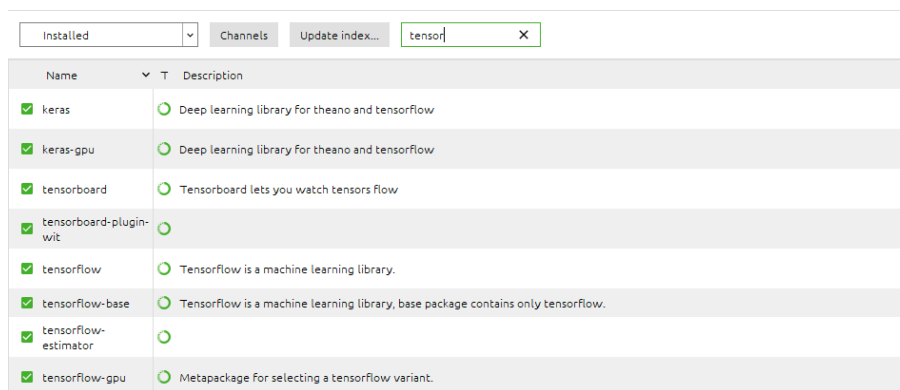


รูปที่ ผ.2 การสร้าง Environment เพื่อใช้งาน โปรแกรมทั้งหมดในการทำวิจัย

### 2.1.3. ติดตั้งไลบรารีที่จำเป็น อย่างน้อยจะต้องมี Tensorflow และ Keras จึงจะสามารถทำงานได้

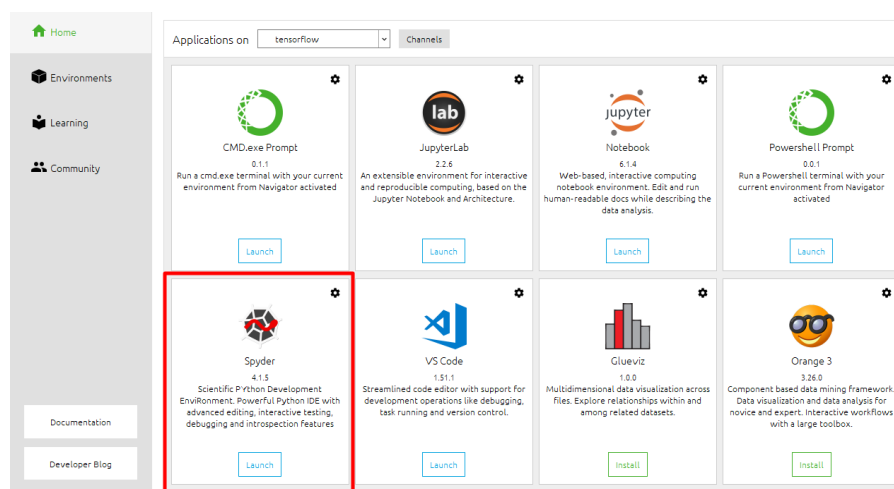


รูปที่ ผ.3 การค้นหาเครื่องมือ Tensorflow และ Keras



รูปที่ ผ.4 รูปไลบรารีที่จำเป็นหลังติดตั้งเสร็จสิ้นแล้ว

### 2.1.3. เมื่อติดตั้งเสร็จ ให้เปิดด้วยโปรแกรม Spyder ผ่านสภาพแวดล้อมที่ Anaconda สร้างเอาไว้



รูปที่ ผ.5 การเปิดแอปพลิเคชัน Spyder ผ่าน Anaconda Navigator

## 2.2. โปรแกรม Packet Generator

### 2.2.1. ทำการแตกไฟล์ Packet Generator.rar

### 2.2.2. กำหนดค่า Parameter ต่างๆที่ใช้ในการสร้างชุดข้อมูล

```
import csv

csv_file_text = "%s.csv" % "train_text"
csv_file_bin = "%s.csv" % "train_binary"
```

รูปที่ ๒.๖ การกำหนดชื่อไฟล์ที่ต้องการ

```
ip_src_all = []
net4 = ipaddress.ip_network('192.168.0.0/16')

for x in net4.hosts():
    ip_src_all.append(str(x))

"""Assign IP Destination Address here"""
ip_dst_all = ['161.246.34.11/24']

"""Assign Port here"""
port_all = ['22', '80']

"""Assign Protocol here"""
protocol_all = ['6', '17']
```

รูปที่ ๒.๗ การกำหนดขอบเขตของ Data Field ที่จะศึกษา

```
# ----- RULES -----
"""Assign Firewall Rule here"""
rule_1 = ['allow', '192.168.0.0/16', '161.246.34.11/24', '80', '6']
rule_2 = ['deny', '192.168.0.0/16', '161.246.34.11/24', '80', '17']
rule_3 = ['deny', '192.168.0.0/16', '161.246.34.11/24', '22', '6']
rule_4 = ['deny', '192.168.0.0/16', '161.246.34.11/24', '22', '17']

# ----- RATIO -----
"""Assign Packet Number Wanted"""
ruleN_1 = 100
ruleN_2 = 100
ruleN_3 = 100
ruleN_4 = 100
default = 0
```

รูปที่ ๒.๘ การกำหนดเงื่อนไขของชุดกฎไฟร์วอลล์และจำนวนข้อมูลในแต่ละกฎ

### 2.2.3. กดคำสั่งเริ่มเพื่อให้โปรแกรมทำงาน

```
def random_packet(): # will fix it later

    src_address = random.choice(ip_src_all)
    src_mask = "255.255.0.0"
    dst_address = str(ipaddress.IPv4Interface(ip_dst_all[0]).ip)
    dst_mask = str(ipaddress.IPv4Interface(ip_dst_all[0]).netmask)
    port = random.choice(port_all)
    protocol = random.choice(protocol_all)

    raw_data_packet = [src_address, src_mask, dst_address, dst_mask, port, protocol]

    return raw_data_packet

def rule_packet_possible(firewall_rule):

    raw_data_packet = []

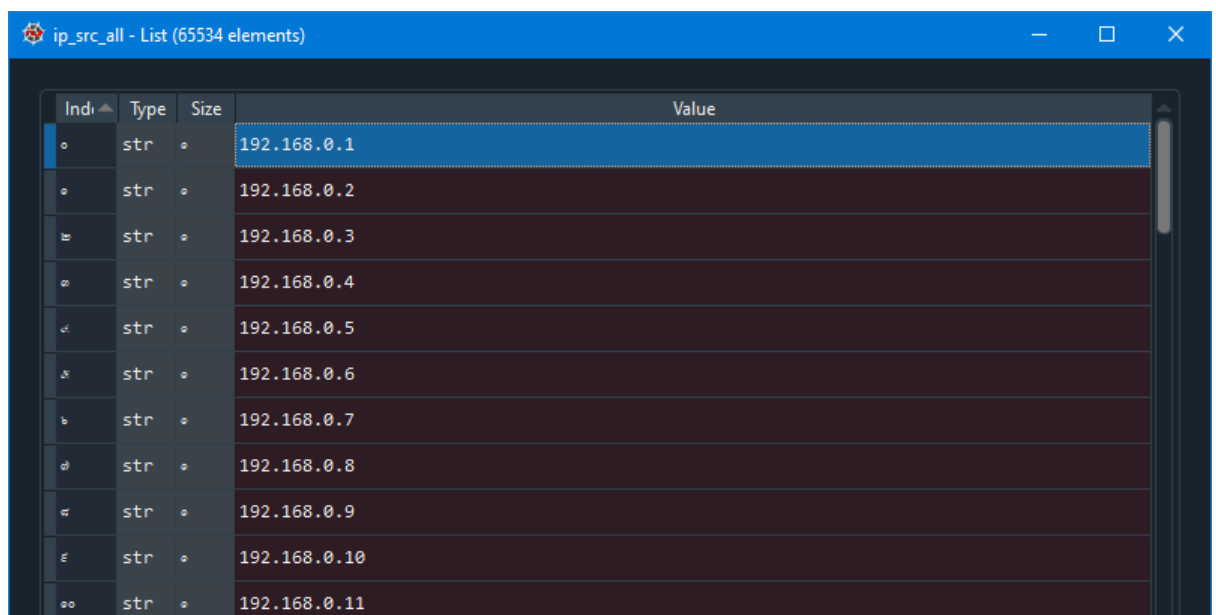
    net4 = ipaddress.ip_network(firewall_rule[1])

    for x in net4.hosts():
        raw_data_packet.append([str(x), '255.255.0.0', str(ipaddress.IPv4Interface(ip_dst_all[0]).ip),
                                str(ipaddress.IPv4Interface(ip_dst_all[0]).netmask), firewall_rule[3], firewall_rule[4]])

    return raw_data_packet
```

รูปที่ ผ.9 โค้ดการทำงานสำหรับการสุ่มชุดข้อมูล

รูปที่ ผ.9 เป็นฟังก์ชันการทำงานโดยการป้อนกฎไฟร์วอลล์เข้าไป แยกส่วนของชุดกฎไฟร์วอลล์มาตีความและสร้างออกมาเป็น List ที่ประกอบไปด้วยชุดข้อมูลที่เป็นไปได้ทั้งหมดของกฎไฟร์วอลล์นั้น โดยจะเก็บเป็นตัวแปรเอาไว้ เพื่อใช้หาชุดข้อมูลที่เป็น Default Rule



Indi	Type	Size	Value
0	str	0	192.168.0.1
1	str	0	192.168.0.2
2	str	0	192.168.0.3
3	str	0	192.168.0.4
4	str	0	192.168.0.5
5	str	0	192.168.0.6
6	str	0	192.168.0.7
7	str	0	192.168.0.8
8	str	0	192.168.0.9
9	str	0	192.168.0.10
10	str	0	192.168.0.11

รูปที่ ผ.10 สร้าง List ที่ประกอบไปด้วยจำนวนข้อมูลที่เป็นไปได้ทั้งหมดในกฎไฟร์วอลล์นั้น

rule\_2\_possible - List (65534 elements)

Indi	Type	Size	Value
๐	list	๖	['192.168.0.1', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๑	list	๖	['192.168.0.2', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๒	list	๖	['192.168.0.3', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๓	list	๖	['192.168.0.4', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๔	list	๖	['192.168.0.5', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๕	list	๖	['192.168.0.6', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๖	list	๖	['192.168.0.7', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๗	list	๖	['192.168.0.8', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๘	list	๖	['192.168.0.9', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๙	list	๖	['192.168.0.10', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๑๐	list	๖	['192.168.0.11', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]
๑๑	list	๖	['192.168.0.12', '255.255.0.0', '161.246.34.11', '255.255.255.0', '80', ...]

รูปที่ ๗.11 ตัวอย่างของชุดข้อมูลที่ได้มาจากการสุ่ม

```
#----- raw train data set from rule -----

rule_1_possible = rule_packet_possible(rule_1)
rule_1_quota = [] # use this as output
for i in range(ruleN_1):
    temp = [rule_1[0]] + random.choice(rule_1_possible)
    rule_1_quota.append(temp)

rule_2_possible = rule_packet_possible(rule_2)
rule_2_quota = [] # use this as output
for i in range(ruleN_2):
    temp = [rule_2[0]] + random.choice(rule_2_possible)
    rule_2_quota.append(temp)

rule_3_possible = rule_packet_possible(rule_3)
rule_3_quota = [] # use this as output
for i in range(ruleN_3):
    temp = [rule_3[0]] + random.choice(rule_3_possible)
    rule_3_quota.append(temp)

rule_4_possible = rule_packet_possible(rule_4)
rule_4_quota = [] # use this as output
for i in range(ruleN_4):
    temp = [rule_4[0]] + random.choice(rule_4_possible)
    rule_4_quota.append(temp)
```

รูปที่ ๗.12 กำหนด List ทั้งหมดที่ประกอบไปด้วยชุดข้อมูลไฟร์วอลล์ที่เป็นไปได้

รูปที่ ผ.12 เป็นการเรียกใช้ฟังก์ชันจาก รูป ผ.9 ซ้ำๆกัน แต่มาจากแต่ละกฎไฟร์วอลล์ ซึ่งในแต่ละกฎจะได้ตัวแปรอีกตัวหนึ่งซึ่งเป็น List ที่ใช้เก็บจำนวนโควต้าของชุดข้อมูลที่จะสร้างขึ้น โดยเราได้กำหนดไว้ให้แต่แรกในรูป ผ.8

Indi	Type	Size	Value
0	list	4	['deny', '192.168.245.249', '255.255.0.0', '161.246.34.11', '255.255.2 ...
1	list	4	['deny', '192.168.233.86', '255.255.0.0', '161.246.34.11', '255.255.25 ...
2	list	4	['deny', '192.168.209.187', '255.255.0.0', '161.246.34.11', '255.255.2 ...
3	list	4	['deny', '192.168.2.21', '255.255.0.0', '161.246.34.11', '255.255.255. ...
4	list	4	['deny', '192.168.84.226', '255.255.0.0', '161.246.34.11', '255.255.25 ...
5	list	4	['deny', '192.168.51.207', '255.255.0.0', '161.246.34.11', '255.255.25 ...
6	list	4	['deny', '192.168.51.161', '255.255.0.0', '161.246.34.11', '255.255.25 ...
7	list	4	['deny', '192.168.49.235', '255.255.0.0', '161.246.34.11', '255.255.25 ...
8	list	4	['deny', '192.168.199.157', '255.255.0.0', '161.246.34.11', '255.255.2 ...
9	list	4	['deny', '192.168.42.178', '255.255.0.0', '161.246.34.11', '255.255.25 ...
10	list	4	['deny', '192.168.195.132', '255.255.0.0', '161.246.34.11', '255.255.2 ...
11	list	4	['deny', '192.168.249.102', '255.255.0.0', '161.246.34.11', '255.255.2 ...

รูปที่ ผ.13 ตัวอย่าง List ที่มีจำนวนชุดข้อมูลฝึกสอนตามโควต้าที่กำหนดไว้ในแต่ละกฎไฟร์วอลล์

```
#----- merge all packet in rule to check default-----
all_rule_possible = rule_1_possible + rule_2_possible + rule_3_possible + rule_4_possible

#----- raw train data set from universe -----
default_quota = []
while True:
    rand = random_packet()
    if len(default_quota) == default:
        break
    elif rand not in all_rule_possible:
        temp = ["deny"] + rand
        default_quota.append(temp)

#----- merge list of all train set -----

data_set_text = rule_1_quota + rule_2_quota + rule_3_quota + rule_4_quota + default_quota
train_set_text = rule_1_quota + rule_2_quota + rule_3_quota + rule_4_quota + default_quota
random.shuffle(train_set_text)
```

รูปที่ ผ.14 คัดกรอง Default โดยข้อมูลต้องอยู่นอกขอบเขตของกฎไฟร์วอลล์ที่กำหนดจากรูป ผ.12

รูปที่ ผ.14 เป็นการรวม List ที่ประกอบไปด้วยชุดข้อมูลที่เข้าเงื่อนไขกฎไฟร์วอลล์ที่กำหนด และเริ่มสุ่มชุดข้อมูลทีมาจาก Default Rule ในส่วนนี้ต้องมีการทำงานเป็นลูป เนื่องจากเราไม่ทราบ ว่าข้อมูลฝึกสอนที่ทำการสุ่มได้ออกมาอยู่ในเงื่อนไขกฎไฟร์วอลล์หรือไม่ ถ้าหากอยู่ในเงื่อนไขก็ทำการสุ่มใหม่ โดยจะทำซ้ำไปเรื่อยๆจนได้ชุดข้อมูลที่อยู่นอกเงื่อนไขตามจำนวนที่กำหนด และรวมเข้ากับโควต้าของชุดข้อมูลฝึกสอน

```
#----- binary convert -----
train_set_binary = []

for train_packet in train_set_text:
    binary_a_packet = []
    if train_packet[0] == 'allow':
        binary_a_packet.append('1')
    else:
        binary_a_packet.append('0')
    for j in range(1, 5):
        ip = train_packet[j]
        list_octet = [bin(int(x)+256)[3:] for x in ip.split('.')]
        binary_a_packet.append(list_octet[0])
        binary_a_packet.append(list_octet[1])
        binary_a_packet.append(list_octet[2])
        binary_a_packet.append(list_octet[3])
    binary_a_packet.append(bin(int(train_packet[5])+65536)[3:])
    binary_a_packet.append(bin(int(train_packet[6])+256)[3:])
    # train_packet[6]
    train_set_binary.append(binary_a_packet)
```

รูปที่ ผ.14 รวมชุดฝึกสอนที่อยู่ในจำนวนโควต้าที่กำหนด ทำเป็นเลขฐานสอง

train_set_text - List (4000 elements)				
Indi	Type	Size	Value	
๐	list	๘	['deny', '192.168.33.154', '255.255.0.0', '161.246.34.11', '255.255.25 ...	
๑	list	๘	['deny', '192.168.0.153', '255.255.0.0', '161.246.34.11', '255.255.255 ...	
๒	list	๘	['allow', '192.168.249.39', '255.255.0.0', '161.246.34.11', '255.255.2 ...	
๓	list	๘	['allow', '192.168.218.215', '255.255.0.0', '161.246.34.11', '255.255. ...	
๔	list	๘	['allow', '192.168.252.10', '255.255.0.0', '161.246.34.11', '255.255.2 ...	
๕	list	๘	['allow', '192.168.167.45', '255.255.0.0', '161.246.34.11', '255.255.2 ...	
๖	list	๘	['allow', '192.168.197.171', '255.255.0.0', '161.246.34.11', '255.255. ...	
๗	list	๘	['deny', '192.168.226.36', '255.255.0.0', '161.246.34.11', '255.255.25 ...	
๘	list	๘	['deny', '192.168.72.27', '255.255.0.0', '161.246.34.11', '255.255.255 ...	

รูปที่ ผ.15 รวมชุดข้อมูลฝึกสอนที่เลือกมาแล้ว ประกอบไปด้วยทุกกฎไฟร์วอลล์ที่กำหนด



Ind	Type	Size	Value
0	list	๑๕	['0', '11000000', '10101000', '00100001', '10011010', '11111111', '111 ...
๑	list	๑๕	['0', '11000000', '10101000', '00000000', '10011001', '11111111', '111 ...
๒	list	๑๕	['1', '11000000', '10101000', '11111001', '00100111', '11111111', '111 ...
๓	list	๑๕	['1', '11000000', '10101000', '11011010', '11010111', '11111111', '111 ...
๔	list	๑๕	['1', '11000000', '10101000', '11111100', '00001010', '11111111', '111 ...
๕	list	๑๕	['1', '11000000', '10101000', '10100111', '00101101', '11111111', '111 ...
๖	list	๑๕	['1', '11000000', '10101000', '11000101', '10101011', '11111111', '111 ...
๗	list	๑๕	['0', '11000000', '10101000', '11100010', '00100100', '11111111', '111 ...
๘	list	๑๕	['0', '11000000', '10101000', '01001000', '00011011', '11111111', '111 ...

รูปที่ ๒.16 แปลงชุดข้อมูลฝึกสอนเป็นเลขฐานสอง

```
#----- csv write -----
with open(csv_file_text, 'w', newline='') as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
    wr.writerow(["Action", "Source address", "Source Mask", "Destination address", "Destination Mask", "Port", "Protocol"])

    for i in train_set_text:
        wr.writerow(i)

with open(csv_file_bin, 'w', newline='') as myfile:
    column = ["Act", "src_a1", "src_a2", "src_a3", "src_a4", "src_m1", "src_m2", "src_m3", "src_m4", "dst_a1", "dst_a2", "dst_a3", "dst_a4"]
    wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
    wr.writerow(column)

    for i in train_set_binary:
        wr.writerow(i)
```

รูปที่ ๒.17 นำชุดข้อมูลฝึกสอนทั้งหมด บันทึกลงในไฟล์ CSV

2.2.4. เมื่อโปรแกรมทำงานเสร็จสิ้น จะได้ไฟล์ชุดข้อมูลนามสกุล .CSV พร้อมรายงานสรุปออกมา

```
In [13]: runfile('C:/Users/POP PC/Documents/GitHub/AI-Firewall-
Training-set-Researching/beta 0.6/1_Packet Gen 4 rule.py', wdir='C:/
Users/POP PC/Documents/GitHub/AI-Firewall-Training-set-Researching/
beta 0.6')
SUMMARY:
Packet Created: 400 packets
Time used: 15.022166013717651 seconds
```

รูปที่ ๒.18 โปรแกรมสร้างชุดข้อมูลรายงานผลสรุปและเวลาที่ใช้

## 2.3. โปรแกรมฝึกโมเดลหรือเครื่องมือโครงข่ายประสาทเทียมเชิงลึก

### 2.3.1. กำหนดตัวแปรต่างๆที่จำเป็นต้องใช้ในการเรียนรู้ของโมเดล

```
# 1 insert local variable here

# File Configuration
csv_file_input = "train_binary" # place the name of data here
csv_file_use = "%s.csv" % csv_file_input

# Model Configuration
node_layer_1 = 150
node_layer_2 = 150
node_layer_3 = 150
epoch = 50

name_model = "model_test" # place the name of model here
name_model_use = "%s.h5" % name_model
```

รูปที่ ผ.19 การกำหนดตัวแปรต่างๆที่ใช้ในการเรียนรู้ของโมเดล

### 2.3.2. กดคำสั่งเริ่มเพื่อให้โปรแกรมทำงาน

```
import pandas as pd
import numpy as py

data = pd.read_csv(csv_file_use)

train_x = data.iloc[:,1:data.shape[1]].values
train_y = data.iloc[:,0].values

train_x = train_x.astype('float32')

import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from keras.optimizers import adam

model = Sequential()

model.add(Dense(node_layer_1, activation='relu', input_shape = (data.shape[1]-1,)))
model.add(Dense(node_layer_2, activation='relu'))
model.add(Dense(node_layer_3, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer="adam", loss='binary_crossentropy', metrics=['accuracy'])
```

รูปที่ ผ.20 โค้ดกระบวนการออกแบบโครงสร้างภายในโมเดล

ในรูปที่ ผ.20 เป็นการตั้งค่าการทำงานและการเรียนรู้ของโมเดล โดยส่วนใหญ่ได้อิงการตั้งค่าแบบ Default และ Rule of Thumb จากปัญญาประดิษฐ์ที่มีข้อมูลและรูปแบบการทำนายที่เหมือนกัน ส่วนที่เป็นการตั้งค่าจะถูกกำหนดไว้ในรูป ผ.19 โดยในส่วนของโค้ดจะเป็นการเรียกใช้งาน โมดูล Keras และออกแบบสร้างโมเดลตามจำนวนโหนดและชั้นที่กำหนด

Index	Act	src_a1	src_a2	src_a3	src_a4	src_m1	src_m2	src_m3	src_m4	dst_a1	dst_a2
0	0	11000000	10101000	10110	1001101	11111111	11111111	0	0	10100001	11110111
1	0	11000000	10101000	1000011	0	11111111	11111111	0	0	10100001	11110111
2	0	11000000	10101000	101	11010011	11111111	11111111	0	0	10100001	11110111
3	0	11000000	10101000	1110000	10000010	11111111	11111111	0	0	10100001	11110111
4	1	11000000	10101000	11001	11100100	11111111	11111111	0	0	10100001	11110111
5	0	11000000	10101000	11001010	11000101	11111111	11111111	0	0	10100001	11110111
6	1	11000000	10101000	11101010	1011010	11111111	11111111	0	0	10100001	11110111
7	0	11000000	10101000	101100	1110100	11111111	11111111	0	0	10100001	11110111
8	0	11000000	10101000	10111100	11011111	11111111	11111111	0	0	10100001	11110111
9	0	11000000	10101000	10011100	1111101	11111111	11111111	0	0	10100001	11110111
10	1	11000000	10101000	1111000	10101100	11111111	11111111	0	0	10100001	11110111
11	0	11000000	10101000	1000000	10001100	11111111	11111111	0	0	10100001	11110111
12	0	11000000	10101000	1011110	11111001	11111111	11111111	0	0	10100001	11110111

รูปที่ ผ.21 List ตัวแปรที่ดึงมาจากไฟล์ CSV ที่ประกอบด้วยชุดข้อมูลฝึกสอน

	0	1	2	3	4	5	6
0	1.1e+07	1.0101e+07	10110	1.0011e+06	1.11111e+07	1.11111e+07	0
1	1.1e+07	1.0101e+07	1.00001e+06	0	1.11111e+07	1.11111e+07	0
2	1.1e+07	1.0101e+07	101	1.101e+07	1.11111e+07	1.11111e+07	0
3	1.1e+07	1.0101e+07	1.11e+06	1e+07	1.11111e+07	1.11111e+07	0
4	1.1e+07	1.0101e+07	11001	1.11001e+07	1.11111e+07	1.11111e+07	0
5	1.1e+07	1.0101e+07	1.1001e+07	1.10001e+07	1.11111e+07	1.11111e+07	0
6	1.1e+07	1.0101e+07	1.1101e+07	1.01101e+06	1.11111e+07	1.11111e+07	0
7	1.1e+07	1.0101e+07	101100	1.1101e+06	1.11111e+07	1.11111e+07	0
8	1.1e+07	1.0101e+07	1.01111e+07	1.10111e+07	1.11111e+07	1.11111e+07	0
9	1.1e+07	1.0101e+07	1.00111e+07	1.11111e+06	1.11111e+07	1.11111e+07	0

รูปที่ ผ.22 ผลลัพธ์การหาค่าน้ำหนักจากการแปลงข้อมูล Data Field

train_y - NumPy object array	
	0
0	0
1	0
2	0
3	0
4	1
5	0

รูปที่ ผ.23 ส่วนของ Field Decision ที่แบ่งออกมาใช้ในการอ้างอิงผลลัพธ์และฝึกสอน

```
import time
print("Training . . . . .")
time_start = time.time()
# Training phase
model.fit(train_x, train_y, epochs = epoch)
# End count training time
time_finish = time.time()
time_duration = time_finish - time_start
```

รูปที่ ผ.24 โค้ดการจับเวลา และการเริ่มโมเดลให้ทำเรียนรู้จากชุดข้อมูล

เนื่องจากเวลาที่ใช้ในการฝึกสอน เป็นผลลัพธ์ที่สำคัญในเชิงเปรียบเทียบประสิทธิภาพ จึงจำเป็นต้องมีการจับเวลาตั้งแต่เริ่มฝึกโมเดล และหยุดจับเวลาเมื่อโมเดลมีการรายงานผลลัพธ์การฝึกสอนโมเดล

```
# Do summary of training
model.summary()

score, acc = model.evaluate(train_x, train_y)

print("Training time:", str(time_duration) + " Seconds")
print('Train score:', score)
print('Train accuracy:', acc)

model.save(name_model_use)
```

รูปที่ ผ.25 โค้ดการรายงานและสรุปผลการเรียนรู้ของโมเดล

2.3.3. เมื่อโปรแกรมทำงานเสร็จสิ้น จะได้โมเดลที่มีไฟล์นามสกุล .h5 พร้อมรายงานสรุป

```
400/400 [=====] - 0s 317us/sample - loss:
1532.9993 - accuracy: 0.7825
Training time: 7.046859502792358 Seconds
Train score: 1532.9993408203125
Train accuracy: 0.7825
```

รูปที่ ผ.26 โปรแกรมรายงานผลการฝึกสอนโมเดลหลังบันทึกโมเดล

2.4. ขั้นตอนการใช้งานโปรแกรมตรวจสอบความแม่นยำโมเดล

2.4.1. กำหนดตัวแปร ที่ประกอบไปด้วยชื่อไฟล์และชุดข้อมูลทดสอบที่สร้างขึ้น

```
# File Configuration

csv_file_input = "test_4rule_bin" # place the name of data here
csv_file_use = "%s.csv" % csv_file_input

name_model = "model_test" # place the name of model here
name_model_use = "%s.h5" % name_model
```

รูปที่ ผ.27 การกำหนดตัวแปรต่างๆที่ใช้ในกระบวนการตรวจสอบโมเดล

2.4.2. กดคำสั่งเริ่มเพื่อให้โปรแกรมทำงาน

```
true_positive = 0
true_negative = 0
false_positive = 0
false_negative = 0

import pandas as pd
import numpy as np

data = pd.read_csv(csv_file_use)

test_x = data.iloc[:,1:data.shape[1]].values
test_y = data.iloc[:,0].values

import keras
from tensorflow.keras.models import load_model

model = load_model(name_model_use)
```

รูปที่ ผ.28 การตั้งตัวแปรและโหลดโมเดลที่จะนำมาทดสอบ

```

import time
time_start = time.time()

# prediction = model.evaluate(test_x)
prediction = model.predict(test_x)

# Compare Reference
for i in range(len(prediction)):

    if round(prediction[i][0]) == int(test_y[i]):
        if round(prediction[i][0]) == 1:
            true_positive += 1
        elif round(prediction[i][0]) == 0:
            true_negative += 1

    elif round(prediction[i][0]) != int(test_y[i]):
        if round(prediction[i][0]) == 1:
            false_positive += 1
        elif round(prediction[i][0]) == 0:
            false_negative += 1

time_finish = time.time()
time_duration = time_finish - time_start

```

รูปที่ ผ.29 การจับเวลา การทำนายผลที่อิงตาม Reference Variant Set

```

# Accuracy
test_accuracy = float(true_positive + true_negative) / len(prediction)
loss_rate = float(false_positive + false_negative) / len(prediction)

print("Number of Packet: ", len(prediction))
print("Compare Time: %.6f seconds" % time_duration)
print("Accuracy of testing: " + str(test_accuracy*100) + " %")
print("Loss rate: " + str(loss_rate*100) + " %")
print("TP:", true_positive, "TN:", true_negative, "FP:", false_positive, "FN:", false_negative)

```

รูปที่ ผ.30 การสรุปผลลัพธ์ความแม่นยำในการทำนายของโมเดล

2.4.3. เมื่อโปรแกรมทำงานเสร็จสิ้น จะได้รายงานสรุปความถูกต้องของโมเดลที่ทำการตรวจสอบ

```

In [12]: runfile('C:/Users/POP PC/Documents/GitHub/AI-Firewall-
Training-set-Researching/beta 0.6/3_Evaluate.py', wdir='C:/Users/POP
PC/Documents/GitHub/AI-Firewall-Training-set-Researching/beta 0.6')
Evaluating . . . . .
Number of Packet: 40000
Compare Time: 1.555670 seconds
Accuracy of testing: 74.9175 %
Loss rate: 25.082500000000003 %
TP: 7450 TN: 22517 FP: 7483 FN: 2550

```

รูปที่ ผ.31 โปรแกรมรายงานผลสรุปความถูกต้องจากการทดสอบโมเดล

Source code ในงานวิจัย: <https://github.com/Kodashi/AI-Firewall-Training-set-Researching-main>