JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 12, pp. 123-132, December 2021
https://doi.org/10.9708/jksci.2021.26.12.123

A Deep Learning Approach with Stacking Architecture to Identify Botnet Traffic

Koohong Kang*

*Professor, Dept. of Information and Communications Eng., Seowon University, Cheongju, Korea

[Abstract]

Malicious activities of Botnets are responsible for huge financial losses to Internet Service Providers, companies, governments and even home users. In this paper, we try to confirm the possibility of detecting botnet traffic by applying the deep learning model Convolutional Neural Network (CNN) using the CTU-13 botnet traffic dataset. In particular, we classify three classes, such as the C&C traffic between bots and C&C servers to detect C&C servers, traffic generated by bots other than C&C communication to detect bots, and normal traffic. Performance metrics were presented by accuracy, precision, recall, and F1 score on classifying both known and unknown botnet traffic. Moreover, we propose a stackable botnet detection system that can load modules for each botnet type considering scalability and operability on the real field.

▸ **Key words**: Botnet, Botnet Detection System, Deep Learning, Convolutional Neural Network, CTU-13 Dataset

[요 약]

봇 넷의 악의적인 행위는 인터넷 서비스 제공자뿐만 아니라 기업, 정부, 그리고 심지어 가정의 일반 사용자에 이르기까지 엄청난 경제적 손실을 끼치고 있다. 본 논문에서는 CTU-13 봇 넷 트래픽 데이터 셋을 사용하여 딥러닝 모델 Convolutional Neural Network(CNN)을 적용한 봇 넷 트래픽 검출에 대한 가능성을 확인하고자한다. 특히 알려진 봇 넷과 알려지지 않은 봇 넷 트래픽에 대해 C&C 서버를 검출하기 위한 봇과 C&C 서버 사이 트래픽, 봇을 검출하기 위한 C&C 통신 이외에 봇이 발생하는 트래픽, 그리고 정상 트래픽을 분류하는 멀티클래스 분류(multi-class classification)를 시도하였다. 성능검증을 위한 지표는 정확도, 정밀도, 재현율, 그리고 F1 점수를 제시하였다. 한편 확장성과 운영을 고려하여 봇 넷 타입 별로 모듈을 적재할 수 있는 스택구조의 봇 넷 검출 시스템을 제안함으로써 실제 네트워크의 적용 가능성을 제시하였다.

▸ **주제어**: 봇 넷, 봇 넷 검출 시스템, 딥러닝, Convolutional Neural Network, CTU-13 데이터 셋

• First Author: Koohong Kang, Corresponding Author: Koohong Kang
*Koohong Kang (khkang@seowon.ac.kr), Dept. of Information and Communications Eng., Seowon University
• Received: 2021. 09. 23, Revised: 2021. 11. 24, Accepted: 2021. 11. 24.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

봇 넷(Botnet)은 마스터(master 혹은 botmaster)에 의해 C&C(Command and Control) 서버 혹은 P2P(Peer-to-Peer) 네트워크를 통한 원격 제어되는 멜웨어(malware)에 감염된 호스트(혹은 봇)들로 구성된 네트워크이다[1]. 다양한 형태의 봇 넷들은 스팸 메일 공격, 분산서비스거부(DDoS : Distributed Denial of Service) 공격, 정보 및 신원 도용(information and identity theft), 그리고 불법적인 자원 사용 등 악의적인 행위를 실행한다. 이러한 봇 넷의 악의적인 행위는 인터넷 서비스 제공자뿐만 아니라 기업, 정부, 그리고 심지어 가정의 일반 사용자에 이르기까지 엄청난 경제적 손실을 끼치고 있다. 특히 모든 e-메일 트래픽의 약 80%가 봇 넷에 의해 만들어지고 있으며 스팸, 피싱(phishing), 그리고 e-메일 멜웨어와 같은 e-메일 위협은 대규모 조직보다는 소규모 조직에서 더욱 자주 발행하는 것으로 보고되고 있다. 또한 상위 금융 트로이잔 (Top Financial Trojans)를 보면 Zbot(17.6%), Trickybot(2.6%), Qakbot(1.8%)등 주로 봇 넷들에 의해 전파되고 있다[2].

1993년 최초로 IRC(Internet Relay Chat) 봇으로 알려진 Eggdrop 이후, 봇 넷은 매우 다양한 변화를 겪으며 계속 발전해 오고 있다. 특히 새로운 봇들은 봇 마스터와 통신하기 위한 새로운 C&C 프로토콜 사용과 중앙 집중 구조가 아닌 복잡한 새로운 구조 채택, 그리고 더욱 강력한 공격 방법으로 진화하고 있다. 따라서 지난 수 십 년간 이들 봇 넷을 검출하기 위한 다양한 연구가 진행되어 왔다[1]. 이들 봇 넷 검출 기법에는 크게 두 가지 카테고리, 즉 시그너처-기반(signature-based)과 이상징후 기반(anomaly-based)이 있다. 시그너처-기반 봇 넷 검출 시스템은 알려진 봇 넷으로부터 시그너처를 먼저 확보하고, 실제 상황에서는 봇들이 발생시키는 트래픽과 이들 시그너처 데이터베이스와 매칭 과정을 통해 봇 넷을 검출한다. 그러나 시그너처가 확보되지 않은 새로운 봇 넷의 출현 혹은 SSL/TLS(Secure Soket Layer/Transport Layer Security)를 이용한 암호화 통신을 하는 봇 넷의 경우, 이들 시스템들은 시그너처 매칭이 불가능하게 된다. 이러한 문제점을 해결하기 위한, 이상징후-기반 봇 넷 검출 시스템은 비정상적인 네트워크 트래픽 혹은 비정상적인 시스템 행위를 감시하여 봇 넷을 검출하게 된다. 그러나 이상징후-기반 검출 시스템은 높은 오탐율(false positive rate)과 미탐율(false negative rate)의 문제점을 보이고 있다[1].

최근 음성인식, 패턴인식, 그리고 컴퓨터 비전 등 다양한 분야에서 매우 성공적인 발전을 이루고 있는 기계학습(machine learning)과 딥러닝(deep learning) 기술을 컴퓨터네트워크 보안 분야에 적용하기 위한 활발한 연구가 이루어지고 있다[3]. 기계학습의 경우, 비정상 트래픽 패턴을 검출하는 분야에 비교적 잘 적용되어 이상징후-기반 검출 시스템으로 응용되고 있다[4,5,6]. 그러나 높은 오탐율로 인한 검출 오류의 문제점을 나타내고 있으며 사용자의 중제와 많은 노력이 필요한 특성(feature) 엔지니어링 과정에 의존해야 하는 단점이 있다[7]. 이러한 제약들로 인해, 네트워크 보안 응용에 신경망 구조의 딥러닝 접근법이 최근 제안되었고 주요 연구 토픽이 되었다.

본 논문에서는 딥러닝 모델 CNN(Convolutional Neural Network)을 활용한 봇 넷 트래픽 검출에 대한 가능성을 확인하고자 한다. CNN 모델은 딥러닝 모델 중 가장 기본 모델로서 이미지 및 음성 인식뿐만 아니라 네트워크 이상징후 검출을 위한 연구에도 많이 활용되고 있다[8]. 이와 같이 기계학습 혹은 딥러닝 모델을 네트워크 보안과 같은 특정 도메인에 활용하기 위한 검증작업을 위해서는 모든 연구자들에게 공개되는 적절한 데이터 셋 확보가 가장 중요하다. 본 논문에서는 2011년 CTU 대학에서 생성한 실제 봇 넷 트래픽 데이터 CTU-13 데이터 셋을 사용한다[9,10]. 대부분의 기존 연구들은 이들 공개된 데이터 셋의 일부만 사용한 검증 결과를 보여주고 있으나, 본 논문에서는 CTU-13 데이터 셋을 변경 없이 그대로 활용하였다. 또한 봇 넷 검출 관련 기존 연구에서는 봇 넷 트래픽과 정상 트래픽을 구분하는 두 개 클래스 분류 (binary class classification) 결과를 보이고 있으나, 본 논문에서는 C&C 서버 검출을 위한 봇과 C&C 서버 사이 트래픽, 봇(혹은 좀비 zombie 컴퓨터)을 검출하기 위한 C&C 통신 이외에 봇이 발생하는 트래픽, 그리고 정상 트래픽을 분류하는 3개 클래스 분류를 목표로 하고 있다. 이러한 목표는 봇 넷을 검출하고 차단하는 과정에서 C&C 서버 트래픽을 검출하여 차단하는 효과는 일반 봇을 검출하여 제거하는 효과와 비교해 엄청난 장점을 제공하기 때문이다. 한편 본 논문에서는 실제 네트워크 환경에서 학습에 의한 딥러닝 기반의 봇 넷 검출 시스템을 효과적으로 운영하기 위한 방안으로서, 봇 넷 타입(예를 들어, CTU-13 데이터 셋을 기준으로 Neris, Rbot, Virut) 별로 검출 모듈을 설치하여 운영하는 단순한 스택구조(stackable architecture)를 제안하였다. 제안된 스택구조는 시그너처가 확보되지 않았거나 혹은 암호화된 통신을 시도하는 봇 넷의 경우에도 해당 봇 넷 트래픽만 확보되면 학습에 의한 최적화된 검출 모듈을 개별적으로 개발하여 추가함으로써 기존의 운영 중인 검출 모듈과 독립적으로 운영할 수 있는 장점이 있다.

서론에 이어, 제2장에서는 CTU-13 데이터 셋을 활용한 기존의 딥러닝 기반의 봇 넷 검출 시스템을 간략히 기술하고, 이들 기존 연구들에서 CTU-13 데이터 셋 사용에 있어 문제점을 지적하고 본 논문과의 차이점을 설명한다. 제3장에서는 본 연구에서 사용하는 CTU-13 데이터 셋을 설명하고, 제4장에서는 CNN 모델을 활용한 봇 넷 트래픽 검출을 위한 실험환경, CTU-13 데이터 셋 활용을 위한 데이터 전처리 과정, 그리고 제안된 봇 넷 트래픽 검출을 위한 CNN 모델을 설명한다. 제5장에서는 CNN 모델을 활용한 봇 넷 트래픽 검출에 따른 성능을 보일 것이며, 마지막으로 제6장에서 결론 및 향후 연구 방향에 대해 기술하였다.

II. Related Work

본 장에서는 먼저 CTU-13 봇 넷 트래픽 데이터 셋을 대상으로 봇 넷 검출을 위해 딥러닝 모델을 적용한 기존 연구에 대해 간략히 소개하고, 본 논문과 이들 기존 연구와의 차이점을 설명함으로써 본 논문의 기여(contributions)를 기술한다.

Maeda et al.[11]은 Multilayer Perceptron(MLP) 모델을 사용하여 CTU-13 데이터 셋의 일부 트래픽(참고문헌 [11]에 지재된 오차행렬(confusion matrix)로부터 계산한 결과, 봇 넷 인스턴스 19,575, 정상 인스턴스 7,508)을 대상으로 정상 트래픽과 봇 넷 트래픽의 검출 정확도 99%를 제시하였다. 그러나 이들 연구가 CTU-13 데이터 셋으로부터 어떤 트래픽을 추출하여 제안된 MLP 모델의 훈련과 시험을 진행했는지는 확인할 수 없다. 한편, 이들은 CTU-13 데이터 셋이 제공하는 pcap 파일로부터 MLP 모델이 사용할 다양한 특성(features : 패킷 크기의 합, 평균, 표준편차 등)들을 자체적으로 계산하여 사용하였다.

Chen et al.[12]는 P2P 기반 봇 넷 검출을 위해 CNN 기반의 딥러닝 모델을 사용하였다. 이들은 CTU-13 데이터 셋에 포함된 P2P 봇 넷 트래픽에 PeerRush[13]로부터 네 가지 P2P 봇 넷 트래픽을 추가한 10,000개의 플로우(flows)를 대상으로 94%의 봇 넷 트래픽 검출 정확도를 보였다. 그러나 이들 논문에서는 딥러닝 모델을 위한 학습과 시험 데이터 셋을 어떻게 구성하였는지 언급이 없다. 한편 이들은 플로우로부터 7개의 특성(플로우의 입/출력되는 패킷의 비율 등)을 자체적으로 계산하여 사용하였다.

Maimo et al.[14]는 네트워크 이상징후 검출을 위해 이상징후 징후 검출(Anomaly Sympton Detection: ASD)과 네트워크 이상징후 검출(Network Anomaly Detection: NAD)로 구성된 2단계 구조 검출 시스템을 제안하였다. 이들은 ASD 구현을 위해 DBN(Deep Belief Network) 혹은 SAE(Stacked AutoEncoders)를 사용하고 NAD를 위해 LSTM(Long Short-Term Memory) 모델을 각각 사용하였다. 한편 이들은 CTU-13 데이터 셋으로부터 144 특성(전체 플로우에서 입/출력 플로우의 비율 등)을 계산하여 사용하였다. 또한 이들은 먼저 CTU-13 데이터 셋을 훈련(training), 교차검증(validation), 그리고 시험(test)으로 분리하여(알려진 봇 넷 트래픽) 봇 넷 검출 성능을 보였으며, 훈련과 교차검증에서 사용되지 않은 봇 넷 트래픽(알려지지 않은 봇 넷 트래픽)을 대상으로 봇 넷 트래픽 검출 성능을 보였다. 알려진 봇 넷 트래픽의 경우, 봇 넷 트래픽의 99.34% 검출이 가능하였으나 정상 트래픽의 18.74%가 봇 넷 트래픽으로 분류됨으로써 오탐률(false positive rate)이 높은 것으로 조사되었다. 알려지지 않은 봇 넷 트래픽의 경우, 평균 정밀도(precision)가 68% 그리고 재현율(recall)이 평균 71% 성능을 각각 보였다.

Nugraha et al.[7]은 CTU-13 봇 넷 트래픽 데이터 셋을 이용하여 네 가지의 딥러닝 모델(CNN, LSTM, hybrid CNN-LSTM, 그리고 Multi-layer Perceptron(MLP))을 적용한 봇 넷 검출 연구결과를 발표하였다. 이들의 연구결과는 알려지거나 혹은 알려지지 않은 봇 넷 트래픽에 대한 딥러닝 모델의 적용 가능성을 보였으며 성능 지표로서 정확도(accuracy), 민감도(sensitivity), 진짜 음성 비율(specificity), 그리고 F1 점수(score) 등을 제시하였다. 그러나 이들은 CTU-13 데이터 셋이 제공하는 트래픽 중에서 봇에 의한 타켓 컴퓨터(victim systems)를 중심으로 입/출력되는 일부 트래픽만 선택적으로 사용하여 봇 트래픽과 정상 트래픽을 분류하는 2진 분류(binary classification)에 대한 성능지표를 제시하였다. 한편 이들은 CTU-13 데이터 셋이 제공하는 특성 이외에 평균 바이트율(average byte rate)과 평균 패킷률(average packet rate) 2개를 추가했으며, 알려지거나 혹은 알려지지 않은 봇 넷 트래픽에 대해 CNN 모델은 99%의 매우 높은 봇 넷 트래픽 검출 정확도를 보였다.

CTU-13 데이터 셋을 활용하여 딥러닝 모델을 적용한 봇 넷 검출에 대한 이들 기존 연구결과를 살펴보면 아래와 같은 공통된 문제점이 존재한다:

1) CTU-13 데이터 셋이 제공하는 트래픽의 일부만 사용하거나, 다른 데이터 셋을 추가하거나, 혹은 자체적으로 재 정의하여 사용
2) CTU-13 데이터 셋이 제공하는 특성을 사용하지 않고 자체적으로 계산된 특성을 사용

3) 정상 트래픽과 봇 넷 트래픽을 구별하는 2진 분류에 따른 성능 지표 제시
4) 딥러닝 모델에 사용된 훈련, 교차검증, 시험 데이터 셋에 대한 설명이 미흡

본 연구에서는 다음과 같은 기존 연구와의 차별성이 존재한다:

1) CTU-13 데이터 셋이 제공하는 트래픽의 변경 없이 모두 사용
2) CTU-13 데이터 셋이 제공하는 기본 특성만 사용
3) 봇과 C&C 서버 사이 트래픽, C&C 통신 이외에 봇이 발생하는 트래픽, 그리고 정상 트래픽을 분류하는 3개의 클래스 분류
4) 알려진 봇 넷 트래픽(교차검증) 및 알려지지 않은 봇 넷 트래픽(시험 데이터는 훈련 시 사용하지 않은 다른 시나리오로부터의 트래픽)에 대한 성능 지표(정확도, 정밀도, 재현율, F1 점수) 제시

한편 기존 연구들은 봇 넷의 종류에 상관없이 하나로 통합된 딥러닝 모델을 탑재한 검출 시스템을 제안하고 있으나 본 논문에서는 봇 넷의 종류에 따라서 별도의 학습과정에 따른 분리된 검출 모듈을 적재하는 스택구조 시스템을 제안하였다.

III. Botnet Data Set CTU-13

CTU 대학에서는 2011년 봇 넷 트래픽 데이터 셋을 구축하기 위해 서로 다른 봇 넷 샘플을 사용해 13개의 시나리오를 구축하고 이에 따른 양방향 넷플로우(NetFlow) 정보를 일반 연구자들을 위해 웹서비스를 통한 csv 형식으로 제공하고 있다[9,10]. 이들 13개 시나리오들의 특징으로는 봇 넷 C&C 통신에 사용되는 다양한 프로토콜(IRC, P2P, 그리고 HTTP)을 모두 포함하고 있으며 봇 넷들이 발생하는 대부분의 공격 패턴(스팸 메일 발송, 클릭 사기(Click Fraud) 수행, 포트 스캔, 서비스거부공격 등)을 반영하고 있다. Table 1은 13개 시나리오에 대한 측정 기간, 패킷 수, 넷 플로우 수, pcap 파일 크기, 봇 넷의 종류, 그리고 봇의 수를 각각 보여준다. 한편 Neris(시나리오 1, 2, 9), Rbot(시나리오 3, 4, 10, 11), 그리고 Virut(시나리오 5, 13) 봇의 경우, 두 개 이상의 서로 다른 시나리오에 반영되어 있음을 확인할 수 있다.

각 시나리오가 제공하는 넷 플로우 파일 내 플로우들은 Fig. 1에서 보는 바와 같이 15개 필드로 구성된다 - StartTime: 플로우 시작 시간, Dur: 플로우 지속 시간, Proto: 프로토콜, SrcAddr: 발신지 IP 주소, Sport: 발신지 포트 번호, Dir: 방향, DstAddr: 목적지 IP 주소, Dport: 목적지 포트 번호, State: 플로우 상태, sToS: 발

Table 1. Amount of data on each botnet scenario[10]

Id	Duration(hrs)	#Packets	#NetFlows	Size(GB)	Bot	#Bots
1	6.15	71,971,482	2,824,637	52	Neris	1
2	4.21	71,8511,300	1,808,123	60	Neris	1
3	66.85	167,730,395	4,710,639	21	Rbot	1
4	4.21	62,089,135	1,121,077	53	Rbot	1
5	11.63	4,481,167	129,833	37.6	Virut	1
6	2.18	38,764,357	558,920	30	Menti	1
7	0.38	7,467,139	114,078	5.8	Sogou	1
8	19.5	155,207,799	2,954,231	123	Murlo	1
9	5.18	115,415,321	2,753,885	94	Neris	10
10	4.75	90,389,202	1,309,792	73	Rbot	10
11	0.26	6,337,202	107,252	5.2	Rbot	3
12	1.21	13,212,268	325,472	8.3	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34	Virut	1

24:31.3	2994.3176	udp	147.32.84.229	13363	<->	194.44.92.158	39231	CON	0	0	18	2246	1659 flow=Background-UDP-Established
24:31.4	66.339653	tcp	137.191.234.98	42677	->	147.32.84.229	443	FSPA_FSPA	0	0	20	2083	1249 flow=Background-TCP-Established
24:31.4	123.54334	tcp	147.32.84.59	37160	->	209.85.149.95	80	FSPA_FSPA	0	0	54	36629	2340 flow=Background-Established-cmpgw-CVUT
24:31.4	3.876098	tcp	147.32.86.208	3835	->	18.7.29.232	80	SRPA_FSPA	0	0	1604	2078970	14782 flow=Background-TCP-Established
24:31.4	3150.0344	udp	147.32.84.59	123	<->	91.189.94.4	123	CON	0	0	16	1440	720 flow=Background-Established-cmpgw-CVUT
24:31.4	0.143256	udp	147.32.84.59	17021	<->	46.250.70.84	61826	CON	0	0	2	133	73 flow=Background-Established-cmpgw-CVUT
24:31.5	5.729112	tcp	147.32.84.165	1042	->	175.6.0.105	80	SRPA_SPA	0	0	9	1092	509 flow=From-Botnet-V49-TCP-WEB-Established
24:31.5	0.368509	tcp	147.32.84.59	36103	->	2.21.103.81	80	FSPA_FSPA	0	0	62	52757	4435 flow=Background-Established-cmpgw-CVUT
24:31.5	123.40424	tcp	147.32.84.59	36104	->	2.21.103.81	80	FSPA_FSPA	0	0	110	111563	4460 flow=Background-Established-cmpgw-CVUT
24:31.5	120.64809	tcp	147.32.84.59	36105	->	2.21.103.81	80	FSPA_FSPA	0	0	67	47247	3423 flow=Background-Established-cmpgw-CVUT
24:31.5	120.64815	tcp	147.32.84.59	36106	->	2.21.103.81	80	FSPA_FSPA	0	0	44	30702	3090 flow=Background-Established-cmpgw-CVUT
24:31.5	0.148354	udp	147.32.84.229	13363	<->	50.72.152.184	23154	CON	0	0	4	3029	60 flow=Background-UDP-Established
24:31.5	2902.8545	udp	50.13.6.253	29801	<->	147.32.84.229	13363	CON	0	0	6	3137	120 flow=Background-UDP-Established
24:31.5	2.534059	udp	110.49.249.3	43901	<->	147.32.84.229	13363	CON	0	0	6	1174	990 flow=Background-UDP-Established

Fig. 1. Field Information of the flows in CTU-13

신지 서비스 타입, dToS: 목적지 서비스 타입, TotPkts: 전체 패킷 수, TotBytes: 전체 바이트 수, SrcBytes: 발신지에서 전송된 바이트 수, Label: 레이블. 한편 레이블은 정상(Normal) 플로우, 봇 과 봇 마스트 사이 C&C 플로우, 봇이 발생하는 C&C 플로우를 제외한 봇 플로우, 그리고 이들 플로우를 제외한 백그라운드 플로우로 구분된다. 이 때 정상 및 봇 넷 트래픽으로 판정이 불가능하거나 레이블링 조건이 부족한 모든 플로우들은 백그라운드 플로우로 레이블링 되었다. 따라서 본 연구에서는 백그라운드 트래픽은 제외되었다.

IV. Botnet Detection Using CNN

1. Experimental Environments

CNN 모델을 설계하고 시험하기 위해, 본 논문에서는 Intel core i7-9700K 3.6GHz 32GB RAM 윈도우즈10 64비트 개인용 컴퓨터를 사용하였다. 또한 NVIDA GeForce RTX 2080 8GB GPU를 사용하여 훈련 시간을 단축시켰다. 파이썬(Python) 개발 플랫폼은 아나콘다 주피터 노트북(Anaconda Jupyter Notebook) 2019.07 버전을 사용하고 데이터 전처리 등 다양한 수치해석을 위해 싸이킷-런(scikit-learn) 버전 0.21.1, 그리고 텐서플로우(Tensorflow) 버전 1.14.0을 사용하였다.

2. Evaluation Scenarios

제3장에서 설명한 바와 같이 CTU-13 데이터 셋은 13개 시나리오로 구성되어 있다. 본 논문에서는 Fig. 2에서 보듯이 CNN 모델 학습(training) 및 교차 검증(Cross-validation)을 위해 시나리오 8(Murlo), 9(Neris), 10(Rbot), 그리고 13(Virut)을 사용하고 학습된 CNN 모델을 적용한 시험(test) 시나리오를 검증을 위해 시나리오 1, 2, 3, 4, 5, 그리고 11을 사용한다. 본 논문에서는 CNN 모델의 학습 과정에서 보여주지 않은 시험 시나리오에 대해서는 “알려지지 않은 봇 넷 트래픽 - 이상징후(anomaly) 검출”이라는 용어를 사용하였다. Table 2는 이들 시나리오 내 전체 플로우 수, 봇 넷 플로우 수, 정상 트래픽 플로우 수, 그리고 C&C 플로우 수를 각각 보여준다. Table 2에서 보듯이, CTU-13 데이터 셋을 제공하는 웹 사이트 자료[10]에 의한 플로우 수는 본 논문에 사용된 플로우 개수와 약간의 차이가 존재한다. 예를 들어 시나리오 1의 경우, 전체 플로우와 정상 플로우 수는 동일하지만 봇 넷 플로우와 C&C 플로우수는 차이가 있음을 확인할 수 있다. 참고로, 본 연구에서는 CTU-13 데이터 셋으로부터 레이블링된 “Botnet”과 “C&C” 정보를 이용하여 단순 카운팅하여 계산하였다. 이러한 차이는 2진 분류를 시도하는 기존 연구에서는 크게 문제가 되지 않으나 봇 넷과 C&C 플로우, 그리고 정상 트래픽을 구분하는 본 연구를 위해서는 향후 추가적인 검증 작업이 필요하다. 그러나 이들 봇 넷과 C&C 플로우 개수를 합하면 비슷한 수준을 보인다. 예를 들어 시나리오 1의 경우, 본 연구에서는 봇 넷과 C&C

Table 2. Distribution of labels in the NetFlows for each scenario in CTU-13 (na: not available)

Scen.	Reference	Total Flows	Botnet Flows	Normal Flows	C&C Flows
1	our test	2,824,636	40,620	30,387	341
	[10]/[7]	2,824,636/na	39,933/2,693	30,387/8,839	1,026/na
2	our test	1,808,122	20,268	9,120	673
	[10]/[7]	1,808,122/na	18,839/14,362	9,120/5,267	2,102/na
3	our test	4,710,638	26,759	116,887	63
	[10]/[7]	4,710,638/na	26,759/24	116,887/27,433	63/na
4	our test	1,121,076	2,528	25,268	52
	[10]/[7]	1,121,076/na	1,719/1,931	25,268/23,731	49/na
5	our test	129,832	877	4,679	24
	[10]/[7]	129,832/na	695/901	4,679/4,679	206/na
8	our test	2,954,230	5,053	72,776	1,074
	[10]/[7]	2,954,230/na	5,052/1,520	72,822/36,625	1,074/na
9	our test	2,087,508	182,014	29,967	2,973
	[10]/[7]	2,753,884/na	179,880/8,686	43,340/16,690	5,099/na
10	our test	1,309,791	106,319	15,847	33
	[10]/[7]	1,309,791/na	106,315/74,907	15,847/13,052	37/na
11	our test	107,251	8,162	2,718	2
	[10]/[7]	107,251/na	8,161/8,164	2,718/2,718	3/na
13	our test	1,925,149	39,467	31,939	536
	[10]/[7]	1,925,149/na	38,791/23,779	31,939/13,199	1,202/na

플로우 수를 합하면 40,961이고 웹 사이트 자료[10]가 제시하는 플로우 수는 40,959가 된다. 그러나 Table 2를 통해 Nugraha et. al.[7]가 사용한 플로우 개수는 CTU-13 데이터 셋과는 매우 큰 차이를 보임을 확인할 수 있다.

Fig. 2에서 보듯이 본 논문에서 사용한 봇 넷의 종류에 따른 훈련 및 시험을 위한 시나리오의 매핑 관계를 확인할 수 있다. 따라서 본 논문에서는 CNN 모델에 대한 학습 및 교차검증에 의한 “알려진 봇 넷 트래픽”에 대한 검출 성능뿐만이 아니라 실제 네트워크에 적용하기 위해 학습에 사용되지 않은 “알려지지 않은 봇 넷 트래픽” 검출에 대한 CNN 모델의 성능을 확인할 수 있는 기회를 제공한다.

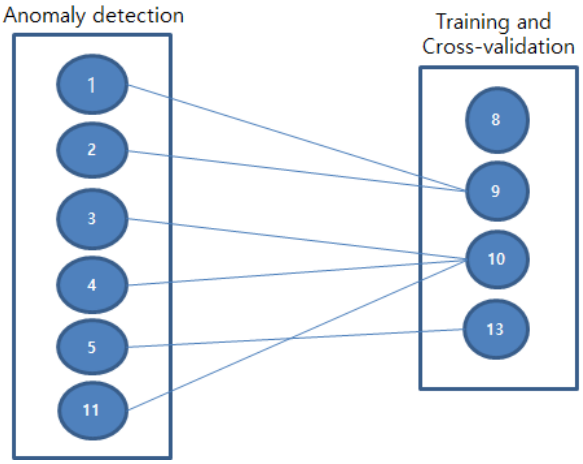

Fig. 2. Evaluation scenarios

3. Dataset preprocessing

CTU-13 데이터 셋은 3장에서 설명한 바와 같이 레이블을 제외한 14개 필드로 구성되어 있으나 본 논문에서는 주소 필드와 CTU-13 데이터 셋이 제대로 처리하지 못해 대부분 ‘0’의 값을 갖는 서비스타입 필드를 제외한 9개의 특성 - Dur(플로우 기간), Proto(프로토콜), Sport(발신지 포트번호), Dir(방향), Dport(목적지 포트번호, State(상태), TotPkts(전체 패킷 수), TotBytes(전체 바이트 수), 그리고 SrcBytes(발신지 바이트 수) - 을 사용한다. 이들 특성 중에서 Fig. 1에서 보는 바와 같이 Proto, Dir, 그리고 State 속성 값은 카테고리 타입에 속한다. 한편 CTU-13 데이터 셋의 일부 인스턴스의 Sport 혹은 Dport 값이 ‘null(널)’ 이거나 혹은 16진수 값(전송계층을 사용하지 않는 인스턴스 들, 예를 들어 ARP 혹은 ICMP 프로토콜)을 가지는 경우가 있다. 따라서 본 논문에서는 이들 Sport와 Dport 특성을 다음과 같이 포트번호 범위를 기준으로 카테고리 변수로 변환하여 사용한다:

- 널 값 : empt (empty)
- 16진수 값 : nosp (not specified)
- 0 ~ 1023 : well (well-known port number)
- 1024 ~ 49151 : regi (registered port number)
- 49152 ~ 65535 : priv (private port number)

Fig. 3은 CTU-13 데이터 셋 시나리오 8 (봇 타입 Murlo)의 카테고리 타입 특성들이 갖는 값들을 모두 나타낸다.

```
category_variables = ["Sport", "Dport", "Proto", "Dir", "State", "Label"]
for cv in category_variables:
    df_result[cv] = df_result[cv].astype("category")

    print("Length of Categories for {} are {}".format(cv , len(df_result[cv].cat.categories)))
    print("Categories for {} are {} \n".format(cv ,df_result[cv].cat.categories))
```

```
Length of Categories for Sport are 4
Categories for Sport are Index(['Snosp', 'Sdriv', 'Sregi', 'Swell'], dtype='object')

Length of Categories for Dport are 4
Categories for Dport are Index(['Dnosp', 'Ddriv', 'Dregi', 'Dwell'], dtype='object')

Length of Categories for Proto are 3
Categories for Proto are Index(['icmp', 'tcp', 'udp'], dtype='object')

Length of Categories for Dir are 4
Categories for Dir are Index(['  ->', '  ?>', '  <->', '  <?>'], dtype='object')

Length of Categories for State are 65
Categories for State are Index(['CON', 'FA_A', 'FA_FA', 'FA_FPA', 'FA_FRA', 'FA_FSA', 'FA_R', 'FPA_FA',
       'FPA_FPA', 'FPA_FRPA', 'FPA_R', 'FPA_RPA', 'FSA_FSA', 'FSA_FSPA',
       'FSA_SA', 'FSA_SPA', 'FSPA_FPA', 'FSPA_FSA', 'FSPA_FSPA', 'FSPA_FSRA',
       'FSPA_FSRPA', 'FSPA_SA', 'FSPA_SPA', 'FSPA_SRA', 'FSPA_SRPA',
       'FSRA_FSA', 'FSRA_FSPA', 'FSRA_FSRA', 'FSRA_SPA', 'FSRA_SRA',
       'FSRPA_FSA', 'FSRPA_FSPA', 'FSRPA_SA', 'FSRPA_SPA', 'INT', 'PA_',
       'PA_A', 'PA_PA', 'PA_RA', 'PA_RPA', 'RA_', 'RA_FPA', 'RPA_PA', 'SA_',
       'SA_SRA', 'SPA_FSPA', 'SPA_FSRPA', 'SPA_SA', 'SPA_SPA', 'SPA_SRA',
       'SPA_SRPA', 'SRA_FSA', 'SRA_FSPA', 'SRA_SPA', 'SRPA_FPA', 'SRPA_FSPA',
       'SRPA_SPA', 'SRPA_SRPA', 'SR_SA', 'S_', 'S_R', 'S_RA', 'URP', '_SA',
       '_SPA'],
      dtype='object')

Length of Categories for Label are 3
Categories for Label are Index(['Bot', 'CC', 'Normal'], dtype='object')
```

Fig. 3. Category types of UTC-13 scenario 8

Fig. 3에서 보듯이, 카테고리 타입 State 특성은 해당 플로우의 양방향 넷 플로우 정보에 대한 연결 상태 및 TCP 플래그 정보를 나타내며 총 64개 종류의 값을 가진다. 이들 State 특성 값은 각 시나리오별 수집된 플로우 특성에 따라 다양한 값을 가질 수 있으며 따라서 각 시나리오별 서로 다른 개수의 값이 존재한다. 또한 각 시나리오 별 지나치게 많은 다양한 State 특성 값을 나타낸다. 이러한 문제점을 해결하기 위해 본 논문에서는 State 특성이 가지는 카테고리 타입의 값의 범위를 축소하고 통일시키기 위해 다음과 같이 TCP 연결의 비정상적인 연결 종료를 감지하기 위해 흔히 사용되는 TCP 플래그[15] 비트 중 리셋 (RST 플래그) 정보를 기준으로 정규화시켜 사용한다.

- 양방향 모두 RST 플래그 존재 : RnR
- 송신측 방향에서만 RST 플래그 존재 : Rn
- 양방향 플래그 값 존재 : n
- 일방향 플래그 값만 존재 : s

Fig. 4는 카테고리 타입 속성들에 대한 데이터 전처리 과정을 거친 후, 최종적으로 시나리오 9 (봇 타입 Neris)

의 카테고리 타입 특성이 갖는 값들을 모두 보여준다. 이와 같은 데이터 전처리 과정을 거친 후(Fig. 5 참조), 카테고리 타입의 속성들은 모두 원-핫 엔코딩(one-hot encoding)[16]을 거쳐 정수 타입의 속성들과 함께 CNN 모델로 입력된다.

```
# Need one-hot encoding for categorical type
#category_variables = ["Sport", "Dport", "Proto", "Dir", "State", "Label"]
# Label (Bot, CC, Normal) 은 y 값, 즉 정답지로 따로 처리함
category_variables = ["Sport", "Dport", "Proto", "Dir", "State"]
# State는 따로 처리할 예정
#category_variables = ["Sport", "Dport", "Proto", "Dir"]
for cv in category_variables:
    df_result[cv] = df_result[cv].astype("category")
    df_test_result[cv] = df_test_result[cv].astype("category")

    print("Length of Categories for {} are {}".format(cv , len(df_result[cv].cat.categories)))
    print("Categories for {} are {} \n".format(cv ,df_result[cv].cat.categories))
```

```
Length of Categories for Sport are 5
Categories for Sport are Index(['Sempt', 'Snosp', 'Spriv', 'Sregi', 'Swell'], dtype='object')

Length of Categories for Dport are 5
Categories for Dport are Index(['Dempt', 'Dnosp', 'Dpriv', 'Dregi', 'Dwell'], dtype='object')

Length of Categories for Proto are 4
Categories for Proto are Index(['arp', 'icmp', 'tcp', 'udp'], dtype='object')

Length of Categories for Dir are 5
Categories for Dir are Index([' ->', ' ?>', ' <->', ' <?>', ' who'], dtype='object')

Length of Categories for State are 4
Categories for State are Index(['Rn', 'RnR', 'n', 's'], dtype='object')
```

Fig. 4. Category types of UTC-13 scenario 9 after pre-processing

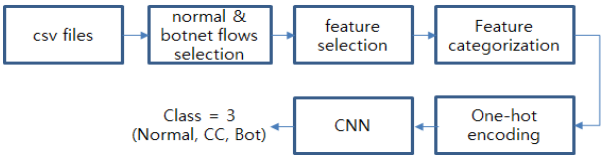

Fig. 5. Data pre-processing for CNN model

4. CNN model

본 논문에서는 Table 3에서 보듯이 매우 다양한 CNN 모델과 하이퍼 파라미터(hyper parameter)를 사용하여 많은 실험을 시도하였다. 본 논문에서는 사전 검토한 이들 실험 결과를 바탕으로 성능 면에서 큰 차이점을 보이지 않는다는 조건하에서 다음과 같은 가장 간단한 CNN 구조를 채택하였다:

- Convolutional Layer : 1 conv2D Layer (20 filters, filter size 2x2, strid 1x1)
- Activation function : ReLU
- Pooling Layer : max pooling in the conv2D layer, strid 2x2
- FCN Layer : 2 Layers (64 and 32)
- Softmax Layer : Output 3
- Others : loss=softmax_cross_entropy, optimizer=Adam, epochs=200, batch_iteration=100, learning_rate=0.01

Table 3. Configurations and hyper-parameters for our test

Hyper-parameter	Values tested
Conv2D layers	1, 2, 3
Filter size, Strid	2x2/1x1, 4x4/1x1, 4x4/1x1
Activation function	ReLU
Pooling Layer	max. strid 2x2
Dropout	0.0, 0.2
FCN layers	1, 2, 3
Learning rate	0.1, 0.01, 0.001

V. Experimental Results

1. Use Case 1: Classifying Traffic from known Botnet

제4장에서 설명한 바와 같이 ITU-13 데이터 셋 시나리오 8(Murlo), 9(Neris), 10(Rbot), 그리고 13(Virut)을 대상으로 "알려진 봇 넷 트래픽 검출" 성능을 확인하다. 이들 네 개 시나리오의 데이터를 각각 훈련과 교차 검증용 데이터 셋 80:20으로 분리하여 학습과 교차검증 과정을 통해 학습을 진행하였다. 본 논문에서 Fig. 6에서 보듯이 이들 봇 넷 타입 별 분리된 봇 넷 검출 모듈을 사용한다. 한편 성능 검증을 위한 시험 데이터 셋은 각 시나리오에서 60%의 인스턴스를 무작위 샘플링하여 구성하였다.

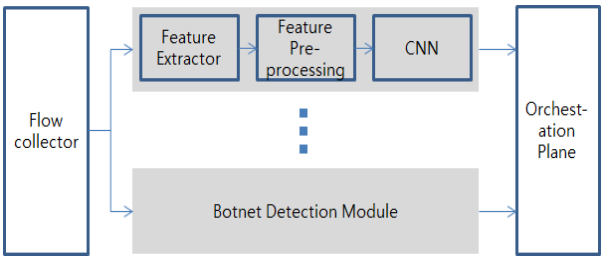

Fig. 6. A Stackable Architecture of Botnet Detection System

제안한 딥러닝 모델 성능을 검증하기 위해 각 시나리오별 Table 4와 같은 오차행렬(confusion matrix)을 구하고 이들로부터 다양한 성능 지표(metric)를 계산할 수 있다. 이때 정확도를 제외한 정밀도, 재현율, 그리고 F1 점수는 각각의 클래스(본 논문에서는 Bot, C&C, 그리고 정상 트래픽)별로 확인하는 것이 가장 정확한 검증작업이 될 것이다. 그러나 너무나 많은 결과 값들을 논문을 통해 모두 확인하는 것은 지면의 제한으로 불가능하며, 따라서 본 논문에서는 정확도를 제외한 이들 지표에 대해서는 매크로 평균값(macro average)을 다음 Fig. 7과 같이 나타내었다. 참고로, 매크로 평균값은 각 클래스별 지표를 단순 평균하여 표

시하지만, 가중치 평균값(weighted average)들은 각 클래스의 인스턴스의 개수에 대한 가중치를 부여해 평균값을 계산한다(Fig. 8참조). 따라서 오차행렬로부터 일부 클래스에 대한 지표가 다른 클래스 지표에 비해 상대적으로 많이 나쁘게 나오면 매크로 평균값(Fig. 7참조)이 가중치 평균값(Fig. 8참조)과 비교해 훨씬 나쁜 성능을 보이게 된다. 예를 들어, Fig. 7로부터 매크로 평균값을 기준으로 시나리오 8(정확도: 99.7%, 정밀도: 99.6%, 재현율: 97.9%, F1 점수: 98.8%)의 성능이 가장 우수하며 시나리오 9(정확도: 92.5%, 정밀도: 87.5%, 재현율: 69.8%, F1 점수: 75.9%)의 성능이 가장 나쁜 것으로 조사되었다. 그러나 Fig. 8에서 보듯이, 이들 두 시나리오의 가중치 평균값의 성능지표들은 상대적으로 큰 차이를 보이지 않는다. 이에 대한 이유는 Fig. 9에서 보듯이, 시나리오 9의 경우는 봇 트래픽에 대한 검출 성능은 매우 우수한 반면 정상 트래픽에 대한 재현율이 상대적으로 매우 낮으며 C&C 트래픽에 대한 정밀도와 재현율이 매우 낮다. 그럼에도 불구하고 Table 4에서 보듯이 검출 성능이 우수한 봇 트래픽의 인스턴스 개수가 매우 많음을 확인할 수 있다. 따라서 Fig. 8과 같이 가중치 평균값을 기준으로 제안한 딥러닝 모델 성능을 평가하면 각 클래스의 인스턴스의 불균형으로 인한 왜곡된 성능지표를 확인하는 오류를 범할 수 있다.

Table 4. Confusion matrices for use case 1

Scen.	Class	Bot	C&C	Normal
8	Bot	2,892	0	120
	C&C	0	630	14
	Normal	12	3	43,699
9	Bot	109,129	90	108
	C&C	709	1,031	13
	Normal	8,470	318	9,105
10	Bot	63,631	3	105
	C&C	0	21	0
	Normal	1	2	9,557
13	Bot	20,063	35	3,487
	C&C	90	193	27
	Normal	578	47	18,646

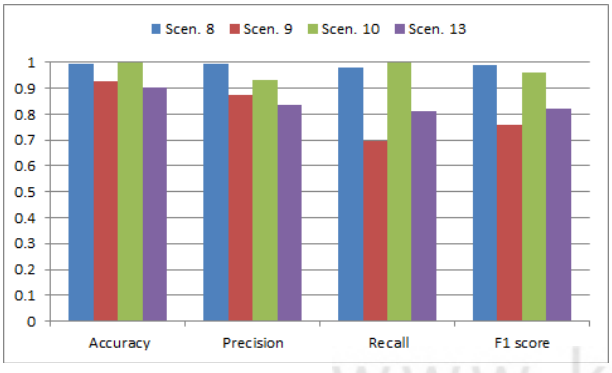

Fig. 7. Results for use case 1 (macro average)

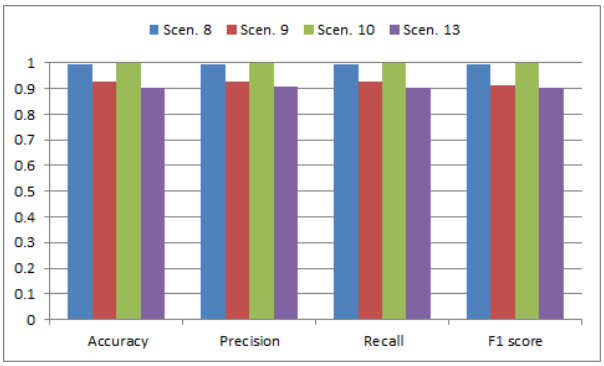

Fig. 8. Results for use case 1 (weighted average)

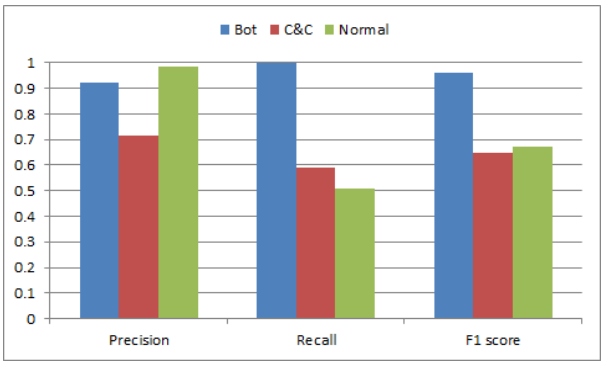

Fig. 9. Scenario 9 result for use case 1

한편, Fig. 7에서 보듯이 정밀도와 재현율의 조화 평균(harmonic mean)을 나타내는 F1 점수를 기준으로 시나리오 8과 10은 시나리오 9와 13과 비교해 상당히 높은 수준의 봇 넷 트래픽 검출 성능을 보인다. 이러한 시나리오 9와 13의 성능저하는 Table 4에서 보듯이 정상 트래픽에 대한 오탐율(false positive)이 높은 사실에 근거하고 있다. 따라서 실제 네트워크에서 제안된 봇 넷 검출 시스템을 사용하기 위해서는 특정 봇 넷 타입에 따른 CNN 이외의 딥러닝 모델 검증 작업 및 ITU-13이 제공하는 특성 이외에 해당 딥러닝 모듈별 특화된 추가적인 특성을 생성해 적용해 할 것이다.

2. Use Case 2: Classifying Traffic from unknown Botnet Traffic

제2장에서 설명한 바와 같이 ITU-13 데이터 셋을 활용한 알려지지 않은 봇 넷 트래픽 검출에 대한 기존 연구 [7]과 [14]에서는 딥러닝 모델의 학습과정에서 사용되지 않은 봇 넷 타입이 전혀 다른 봇 넷 트래픽을 검출하는 과정을 보였다. 그러나 본 논문에서는 제4장에서 설명한 바와 같이 동일한 봇 넷 타입에 대해 학습과정에서 사용되지 않은 전혀 다른 시나리오에 대한 봇 넷 트래픽을 검출하는 과정을 보인다. 즉 5.1절 use case 1에서 학습된 ITU-13 데이

터 셋 시나리오 9(Neris), 10(Rbot), 그리고 13(Virut)에 대해 학습 시 사용되지 않은 시나리오 1(Neris), 2(Neris), 3(Rbot), 4(Rbot), 11(Rbot), 그리고 5(Virut)에 대한 봇 넷 트래픽 검출 성능을 확인한다(Table 5 참조). 오차행렬 Table 5에서 첫 번째 열은 시험 시나리오와 훈련 시나리오를 각각 표시하였다. 즉 첫 번째 행의 1/9는 CNN 모델의 훈련 데이터 셋으로 시나리오 9를 사용하고 시험 데이터 셋으로는 시나리오 1을 사용한 경우이다.

Table 5. Confusion matrices for use case 2 (The first column: Test scenario/Training scenario)

Scen./Scen. (Bot Type)	Class	Bot	C&C	Normal
1/9 (Neris)	Bot	40,364	100	156
	C&C	209	112	20
	Normal	16,178	336	13,873
2/9 (Neris)	Bot	19,174	42	1,052
	C&C	605	62	6
	Normal	4,955	1	4,164
3/10 (Rbot)	Bot	3	0	26,756
	C&C	0	63	0
	Normal	13	4	116,870
4/10 (Rbot)	Bot	1,304	5	1,219
	C&C	0	52	0
	Normal	25	100	25,143
11/10 (Rbot)	Bot	8,146	1	15
	C&C	0	2	0
	Normal	0	0	2,718
5/13 (Virut)	Bot	748	15	114
	C&C	17	4	3
	Normal	188	35	4,456

Fig. 10은 Table 5의 오차행렬로부터 정확도, 정밀도, 재현율, 그리고 F1 점수를 각각 보여주며, 봇 넷의 형태에 따라 봇 넷 트래픽 검출 성능의 차이가 매우 큰 것으로 확인되었다. Neris 봇 넷의 경우는 전반적으로 모든 성능지표가 낮으며 특히 정상 트래픽의 대한 오탐율이 매우 높은 것으로 조사되었다(Table 5 참조). 그러나 Rbot 봇 넷의 경우는 C&C 트래픽에 대한 재현율이 100%이며 정상 트래픽에 대한 재현율이 매우 높음에도 불구하고 봇 트래픽에 대한 검출율(true positive rate)이 다소 낮은 것으로 조사되었다. 한편, Virut 봇 넷의 경우는 이들 두 딥 러닝 모듈 Neris와 Rbot의 중간정도의 성능을 보이는 것으로 분석된다. 결국 알려지지 않은 봇 넷 트래픽에 대한 시험 결과, 정확도 성능지표로는 모두 70% 이상의 성능을 보이고는 있으나 정밀도와 재현율 지표를 기준으로 판단하면 Rbot 봇 넷 모듈을 제외하고는 실제 네트워크에 적용하여 사용하기에는 다소 부족한 측면이 있다. 그러나 본 연구에서는 CNN 모델에 한정하여 실험을 진행하였지만, 추후 각각의 봇 넷 타입에 따른 특화된 딥러닝 모델을 적용할 필요가 있다. 향후, 이와 같이 서로 다른 다양한 딥러닝 모델을 하나의 검출 시스템에 설치해 사용하기 위해서는 본 논문에서 제시한 스택구조의 검출 시스템이 불가피하다.

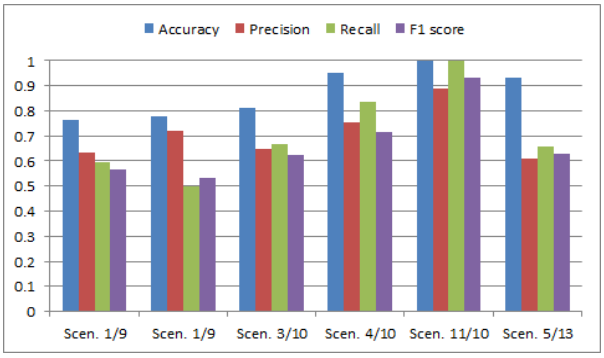

Fig. 10. Results for use case 2 (macro average)

VI. Conclusions

최근 봇 넷 트래픽 검출을 위해 기계학습 혹은 딥러닝 기술을 적용하기 위한 연구가 매우 활발하게 진행되고 있다. 본 논문에서는 봇과 정상 트래픽을 분류하는 기존 연구와는 다르게 봇, C&C, 그리고 정상트래픽을 분류하는 멀티클래스 분류를 위한 스택구조의 봇 넷 검출 시스템을 제안하였다. 모든 봇 넷 트래픽을 통합하여 하나의 학습된 모델을 적용해야 하는 기존 검출 시스템들과 비교해 제안된 스택구조의 검출 시스템은 확장성과 운영성에 있어 실제 네트워크에 적용 가능한 구조로 판단된다. 즉 새로운 봇 넷이 출현하게 되면 해당 봇의 트래픽으로 학습된 새로운 딥러닝 모듈을 기존 모듈에 단순 추가하여 적용함으로써 봇 넷 트래픽 검출 시스템의 실시간 운영을 가능하게 한다.

본 논문에서는 알려진 봇 넷 트래픽과 학습에서 사용되지 않은 봇 넷 트래픽에 대한 알려지지 않은 봇 넷 트래픽 검출 성능을 다양한 성능지표를 통해 확인하였다. 본 논문에서는 CNN 모델만 적용해 실험을 진행하였으며 봇 넷의 타입에 따라서 다양한 성능을 보였다. 하지만, 실제 네트워크에 적용하기에는 일부 봇 넷 타입에 대해서는 정상 트래픽에 대한 오탐율이 지나치게 높은 것으로 조사되었다. CTU-13 데이터 셋을 활용한 기존의 연구들에서는 이러한 성능 저하 문제점을 해결하기 위해 CTU-13이 제공하는 특성 이외에 다양한 특성을 추가해 사용하였으며 제공되는 트래픽의 일부만 선택하여 실험을 진행하였다. 따라서 CTU-13이 제공하는 특성만 사용하고 데이터 셋의 변경 없이 모두 사용하며 그리고 멀티클래스 분류를 시도하는 본 논문의 봇 넷 트래픽 검출 성능과 이들 기존 연구 결과

들과 직접 비교한다는 것은 현실적으로 불가능하다. 결국 본 논문을 통해 제시된 실험 방법과 성능지표는 CTU-13 데이터 셋을 활용한 봇 넷 트래픽 검출을 위한 향후 연구에 좋은 참고가 될 것으로 확신한다.

한편, 본 논문에서는 스택구조에 적용되는 봇 넷 타입 별 동일한 딥러닝 모델 CNN을 사용하였지만 이들 봇 넷 타입 별 서로 다른 딥러닝 모델(CNN, LSTM, CNN-LSTM, MLP 등)을 적용하고 각 모듈 별 서로 다른 특성(features)을 자체적으로 생성하여 사용함으로써 제안된 스택구조의 봇 넷 검출 시스템 최적화 작업을 현재 진행하고 있다.

REFERENCES

[1] S. Silva, R. Silva, R. Pinto, and R. Salles, "Botnets: A survey," Computer Networks, Vol. 57, No. 2, pp. 378-403, 2013, doi:10.1016/j.comnet.2012.07.021

[2] B. O'Gorman, C. Wueest, D. O'Brien, G. Cleary, H.. Lau, J.P. Power, M. Corpin, O. Cox, P. Wood, and S. Wallace, Symantec Internet Security Threat Report, Technical Report, Vol. 24, 2019.

[3] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F.E. Alsaadi, " A Survey of Deep Neural Network Architectures and Their Applications," Neurocomputing, Vol. 234, pp. 11-26, April 2017, doi:10.1016/j.neucom.2016.12.038

[4] M. Tavallaee, E. Bagheri, W. Lu, and A.A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Proceddings of the 2009 IEEE Symposium on Computational Intelligence, pp. 1-6, July 2009, doi:10.1109/cisda.2009.5356528

[5] C. Livadas, R. Walsh, D. Lapsley, and W.T. Strayer, "Using Machine Learning Techniques to Identify Botnet Traffic," Proceedings of the 31st IEEE Conference on Local Computer Networks, 2006, doi:10.1109/lcn.2006.322210

[6] C. Hung and H. Sun, "A Botnet Detection System Based on Machine-Learning using Flow-Based Features," SECURWARE 2018: The Twelfth International Conference on Emerging Security Information, Systems and Technologies, 2018.

[7] B. Nugraha, A. Nambiar, and T. Bauschert, "Performance Evaluation of Botnet Detection using Deep Learning Techniques," Proceedings of the 11th International Conference on Network of the Future, Oct. 2020, doi:10.1109/nof50125.2020.9249198

[8] L. Mohammadpour, T.C. Ling, C.S. Liew,and C.Y. Chong, "A Convolutional Neural Network for Network Intrusion Detection System," Proceedings of the APAN- Research Workshop, 2018.

[9] S. Garcia, M. Grill, J.Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," Computers and Security Journal, Vol. 45, pp. 100-123, 22014, doi:10.1016/j.cose.2014.05.011

[10] The CTU-13 Dataset. A Labeled Dataset with Btnet, Normal and Background traffic, https://www.stratosphereips.org/datasets-ctu13

[11] S. Maeda, A. Kanai, S. Tanimoto, T. Hatashima, and K. Ohkubo, "A Botnet Detection Method on SDN using Deep Learning," Proceedings of 2019 IEEE International Conference on Consumer Electronics, pp. 1-6, 2019, doi:10.1109/icce.2019.8662080

[12] S.C. Chen, Y.R. Chen, and W.G. Tzeng, "Effective botnet detection through neural networks on convolutional features," Proceedings of the 17th IEEE Conference On Trust, Security, And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering, pp. 372-378, 2018, doi:10.1109/trustcom/bigdatase.2018.00062

[13] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Ki, "Peerrush: Mining for Unwanted p2p traffic," Journal of Information Security and Applications, Vol. 19, No. 3, pp. 194-208, 2014, doi:10.1007/978-3-642-39235-1_4

[14] L.F. Maimo, A.P. Gomez, F.G. Clemente, M.G. Perez, and G.M. Perez, "A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks," IEEE Access, Vol. 6, pp. 7700-7712, 2018, doi:10.1109/access.2018.2803446

[15] W.R. Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley, 1994.

[16] A. Geron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'REILLY, 2017.

Authors

Koohong Kang received the B.S. and M.S. degrees in Electronics Engineering from Kyungpook National University and Chungnam National University, Korea, in 1985 and 1990, respectively, and his PhD degree in 1998 in Computer Science and Engineering from Pohang University Science and Technology(POSTECH), Korea. He joined the faculty of the Department of Information and Communications Engineering at Seowon University, Cheongju, Korea, in 2000. He is interested in network security and artificial intelligence.