J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

CNN을 이용한 DNS DRDoS 탐지방법에 관한 연구☆

Research on DNS DRDoS Detection Methods Using CNN

신 훈[1] 정 재 영[1] 조 규 민[1] 이 재 일[2] 신 동 규[1,3*]
Hoon Shin Jaeyeong Jeong Kyu-min Cho Jae-il Lee Dong-kyoo Shin

요 약

DNS DRDoS 공격은 DDoS 공격 기법 중 하나로, 여러 대의 PC에서 공격 대상 시스템의 소스 IP를 위조하여 DNS 서버에 요청을 보내고, 이에 대한 응답 패킷이 대량으로 공격 대상 시스템으로 향하게 하는 공격 방식이다. 이러한 공격은 발신지를 속이는 특성 때문에 공격자의 식별이 어렵고, UDP 프로토콜을 사용하여 TCP 기반 DDoS 공격과 달리 세션 기반의 이상 탐지가 어렵다는 특징이 있다. 본 연구에서는 DNS DRDoS 공격을 탐지하기 위한 새로운 접근 방식으로, 공격 패킷을 이미지화하여 딥러닝의 한 기법인 합성곱 신경망(CNN)으로 학습시키는 방법을 제안한다. 이를 통해 DNS DRDoS 공격 탐지의 정확도를 평가하였으며, 실험 결과, 정확도 99.81%를 확인하였다. 본 연구는 기존의 탐지 기법과 비교하여 메타데이터 추출 등 전처리 과정이 없이도 신속하고도 높은 정확도를 보여 네트워크 보안 분야에서 딥러닝을 이용한 효과적인 DNS DRDoS 방어체계를 제시하였다.

☞ 주제어 : AI, AI보안, 인공지능, 머신러닝, 기계학습, 딥러닝, CNN, 디도스, 분산증폭공격, 이미지 분류, 이미지 처리

ABSTRACT

Domain Name System (DNS) amplification Distributed Reflection Denial of Service (DRDoS) is a type of Distributed Denial of Service (DDoS) attack in which multiple hosts spoof the source IP to that of the target system and send requests to DNS servers. As a result, the response packets flood the target system. The attacker conceals the origin of the attack, making it difficult to identify the attacker or detect abnormal packets. Additionally, legitimate DNS servers are often used as attack agents. Since User Datagram Protocol (UDP) is used in these attacks, it is challenging to detect anomalies based on session protocols, as is done in Transmissoon Control Protocol (TCP)-based DDoS attacks. In this paper, we propose a method for detecting DNS amplification DRDoS attacks by converting attack packets into images and training a Convolutional Neural Network (CNN) to recognize these attacks. Although this method may have a lower detection rate compared to approaches that extract and learn specific DDoS characteristics from packets, it offers the advantage of faster detection due to the omission of the data preprocessing step. Given the nature of DDoS attacks, where real-time response is often more critical than achieving near-perfect detection accuracy, this faster detection capability can be particularly valuable in practical scenarios.

☞ keyword : AI security, artificial intelligence, Machine Learning, Deep Learning, CNN, DDoS, DRDoS, Image processing, Image Classification

1. 서 론

DRDoS(Distributed Reflection Denial of Service) 공격은 공격자가 타겟 시스템의 IP 주소를 위조해 여러 서버에 요청을 보내고, 그 서버들로부터 생성된 대규모 응답 패킷을 타겟 시스템으로 유도해 네트워크를 마비시키는 방식이다. 이 공격은 대규모 서비스 중단을 초래해 기업에 심각한 경제적 손실과 신뢰도 하락을 가져올 수 있다. 게다가 공격자의 위치를 숨기기 쉬워 추적이 어렵고 네트워크의 취약점과 결합하여 악용되기도 하는 등 네트워크 보안에서 중요한 문제로 다뤄진다.

DRDoS 공격의 위협은 여러 실제 사례를 통해 입증되었다. 2013년 3월, 비영리 스팸대응 조직인 스팸하우스(Spamhaus)를 대상으로 최대 300Gbps에 이르는 DNS DRDoS 공격이 발생하여 큰 피해를 입혔으며[1], 2016년 10월에는 미국의 주요 DNS 서비스 제공업체인 Dyn이 수백만 대의 IoT 장치로 구성된 미라이(Mirai) 봇넷을 이용

1 Department of Computer Engineering, Sejong University, Seoul, 05006, Korea.
2 CISO, Smilegate, Gyeonggi-do, 13493, Korea
3 Department of Convergence Engineering for Intelligent Drones, Sejong University, Seoul, 05006, Korea.
* Corresponding author (shindk@sejong.ac.kr)
[Received 09 October 2024, Reviewed 22 October 2024, Accepted 04 November 2024]
☆ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2022R1F1A1074773). 이 논문은 2023년도 세종대학교 교내연구비 지원에 의한 논문임.

한 대규모 DDoS 공격을 받아 Twitter, Netflix, Reddit, GitHub 등 여러 유명 사이트가 접속 불가능한 상태에 빠지기도 했다[2]. 2018년 2월에는 GitHub가 Memcached 서버를 악용한 DRDoS 공격의 대상이 되어 초당 약 1.35Tbps의 트래픽을 받았고, 이는 GitHub의 DNS 서버에 심각한 과부하를 일으켰다[3]. 최근 2024년 1분기 Cloudflare 보고서[4]에 따르면, DNS DRDoS 공격은 전년대비 80% 증가하며 그 위협이 여전히 지속되고 있음을 보여준다. 이러한 사례들은 DNS DRDoS 공격의 파괴력과 그에 대한 지속적인 경계의 필요성을 잘 보여준다.

DRDoS 공격은 TCP 서비스를 대상으로 할 수도 있지만 방어자에게 프로토콜상의 비정상 정보를 최소화하고 탐지를 어렵게 하기 위해 ICMP, UDP 등 Connectionless 기반 프로토콜을 선호한다. SNMP, NTP, CHARGEN 등 여러 UDP 서비스 중에서도 필수 서비스로 포트 차단이 불가능한 DNS 서비스가 자주 공격의 대상이 되고 있다. 따라서 DNS DRDoS의 경우 대부분 시간대별 트래픽 양을 기반으로 DDoS 탐지를 수행하게 되는데 이는 사용자가 지정한 일정한 임계값(threshold)를 넘지 않으면 이 또한 탐지가 어렵게 된다. 공격자들은 초기에 적은 패킷을 보내서 테스트를 하기도 하고 점진적으로 트래픽 양을 조정하는 등 탐지를 방해한다.

기존의 인공지능을 이용한 DDoS 탐지 연구들은 탐지율을 높이기 위해 패킷의 주요 메타데이터를 추출하여 기계학습이나 딥러닝으로 탐지하는 방식이었으며 이를 위해서는 탐지율을 높이기 위해 상당한 데이터 전처리가 필요했다.

본 연구에서는 패킷을 단순 이미지화 한 후, 이미지 딥러닝 기법인 CNN(Convolutional Neural Network)을 기반으로 하여 DDoS의 한종류인 DNS DRDoS(DNS Amplication Distributed Reflection Denial of Service) 공격을 탐지해보는 연구접근 방식을 제안하고 이를 실험적으로 검증하였다.

본 연구의 주요 기여는 다음과 같다. 첫째, 기존의 머신러닝, 딥러닝 기반 DDoS 탐지 방법들이 주로 패킷의 메타데이터를 추출 등 전처리를 통해 탐지율을 높이려는 반면, 본 연구에서는 패킷을 단순 이미지화하여 전처리 과정 없이 신속하게 탐지가 가능하게 하였다. 이는 실시간 대응이 중요한 DDoS 공격 탐지 상황에서 데이터 전처리 시간을 크게 줄일 수 있는 장점을 제공한다. 둘째, 가장 효율적인 CNN모델 설계를 위하여 각종 매개변수를 수정해가며 실험을 하였으며 이를 통해 가장 적절한 안정성과 성능을 보이는 모델을 제안한다. 셋째, CNN을 활용한 이미지 기반 딥러닝 탐지모델을 제안하고 머신러닝 기법들과 비교하여 전처리 없는 이미지 데이터로도 높은 탐지정확도를 유지하는 새로운 접근 가능성을 제시하였다.

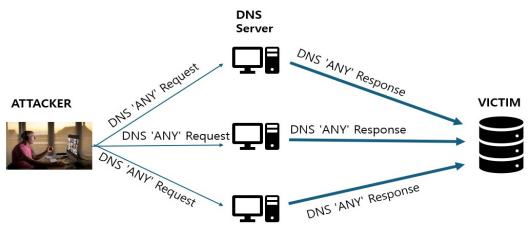
2. 관련 연구

2.1 DNS DRDOS

DRDoS(Distributed Reflection DoS)는 다수의 시스템에서 소스 IP를 공격하고자 하는 시스템 IP로 지정(spoofing)하여 발송하여 응답패킷이 공격대상 시스템으로 집중되게 하는 공격 기법이다. 공격자가 발신지를 속여서 보내어 누가 공격하였는지 알기 어렵고 기존의 정상적인 서버들이 공격 에이전트, 즉 공격 경유지가 되는 특징이 있다. 서버들은 응답패킷을 공격대상 시스템에 전송하게 되고 공격자가 지정한 경유지 서버들은 다수의 응답패킷을 생성하여 대상 시스템에 서비스 거부 현상을 일으키는 방식이다. 한편, DRDoS 공격중 응답패킷이 발송패킷에 비해 증폭되는 기법을 사용하여 트래픽 규모를 증가시키는 기법을 DRDoS Amplication 공격이라고도 한다.

DRDoS 공격으로는 ICMP Flooding, TCP Flooding, DNS Amplification(Reflection) Flooding, NTP Amplification Flooding, SNMP Amplification Flooding, SSDP Amplification Flooding, Memcached Amplification Flooding 등이 있으며 이중 DNS Amplication DRDoS의 경우, 그림 1에서 보듯이 공격자는 Victim Host로 위장하여 여러 Recursive DNS Server에 요청 패킷을 보낸다. 그러면 DNS Server들은 Victim Host에 응답 패킷을 보내는데 이 응답 패킷은 요청 패킷에 비해 커질 수 있기 때문에 Victim Host에 서비스 거부 현상 발생시킨다.

DNS 서버에 질의시에 레코드값을 ANY type으로 지정하면 DNS 서버는 질의 도메인과 연관된 모든 레코드 정보를 보내주게 되며 이는 증폭의 효과를 내게 된다. HJ Kim 등[5]에 따르면 264~499%의 증폭이 관찰되었다.



(그림 1) DNS 증폭 DRDOS 동작과정
(Figure 1) DNS amplication DRDOS process

2.2 딥러닝 기반의 트래픽 분석 연구

Aydın 등[6] 은 클라우드 환경에서 실시간 DDoS 대응이 가능하도록 경량화한 LTML 모델을 제안하여 98.2%의 정확도와 97.5%의 탐지율을 기록했으며 SVM과 Decision Tree기반 기존 ML모델과 비교했을 때, 탐지율과 F1 스코어가 더 우수함을 보여주었다.

Diaba과 Elmusrati [7] 는 스마트 그리드 환경에서 DDoS 탐지를 고려한 경량화된 CNN과 LSTM을 결합한 모델을 제안, 98.4%의 정확도를 기록했으며 Random Forest 및 KNN과 비교, 약 5% 이상 성능 향상을 나타내었다.

Gadze 등[8] 은 DDoS 탐지를 위한 딥러닝모델을 개발, CNN-LSTM 하이브리드 모델로 각각 99.4%, 98.7%의 정확도와 재현율을 보여주었고, SVM, Logistic Regression, 그리고 단일 CNN 모델과 비교하여 2% 이상 높은 정확도를 달성하였다.

Singh과 Jang-Jaccard[9] 는 비지도 학습인 이상트래픽 탐지에 CNN과 LSTM, AE 모델을 결합하여 96.9%의 정확도, 95.4%의 F1 스코어를 기록하였으며 Autoencoder와 CNN 기반 모델보다 약 4% 이상의 우수한 성능을 기록하였다.

Aswad 등[10]은 IoT 네트워크에 발생하는 DDoS 탐지를 위해 경량화된 CNN-LSTM 모델을 개발하여 97.6%의 정확도와 96.8%의 탐지율을 기록하였으며 SVM과 Decision Tree 모델 대비, 6% 이상의 성능 향상이 있었다.

Ahmed 등[11]은 CNN과 RNN을 결합한 하이브리드 모델을 사용하여 네트워크 패턴을 학습하고 DDoS 공격을 탐지 하였으며 정확도 98.3%, 탐지율 97.6%로 단일 CNN 또는 RNN 모델보다 성능이 3% 이상 향상되었다.

2.3 이미지화 기반의 트래픽 분석연구

I. H. Seo 등[12] 은 DNS DRoS 탐지를 위해 약 400개의 네트워크 패킷들에서 메타데이터를 추출, 압축하여 21x21의 2차원 이미지로 표현하여 CNN으로 학습하였으며 98.55%의 정확도를 나타내었다.

K. M. Jeong[13] 은 네트워크 트래픽을 이미지화하여 딥러닝에 적합한 전처리기법을 제안했는데 세션 단위로 읽고(종료 조건은 FIN flag나 RST flag를 수신하거나 마지막 패킷 수신이후 30초 이내에 추가적인 패킷이 없을 때까지)이를 txt파일로 저장하고 N×N 이미지 파일 PNG 형태로 변환하는 것을 제안하였다.

Kim T 등[14]은 네트워크 트래픽을 일반적인 흑백 이미지로 변환하지 않고 3채널 RGB 컬러 이미지로 변환하여 딥러닝을 이용한 침입탐지 성능을 향상시켰다.

컬러와 흑백 이미지를 이용한 딥러닝 모델의 성능 비교연구와 관련, J. Kim 등[15] 은 DoS-Hulk, DoS-Slow HTTPTest, DoS-GoldenEye, DoS-Slowloris, DDoS-LOIC-HTTP, DDoS-HOIC 등 고급 DoS 공격을 포함한 데이터셋인 CSE-CIC-IDS 2018을 RGB와 흑백이미지로 생성하여 CNN 모델로 실험한 결과, 이진 및 다중 클래스 분류 모두에서 RGB 이미지가 흑백 이미지보다 더 높은 정확도를 보였다. CNN 모델은 평균 91.5%의 정확도를 보인 반면, RNN 모델은 평균 65%의 정확도를 보였다.

Y. He 등[16] 은 암호화된 트래픽 분류에 있어서 암호화된 트래픽을 흑백 이미지로 변환하고, 변환된 흑백이미지를 CNN으로 분류하여 암호화된 네트워크 트래픽을 분류하였으며 기존의 암호화된 트래픽 분류에 대해 F1 점수 97.73%, 가상 사설망(VPN) 분류에 대해 F1 점수 99.55%를 달성하였다.

이처럼 트래픽을 이미지화 한후 기계학습과 딥러닝 모델을 통하여 네트워크의 이상징후탐지, DDoS 탐지, 트래픽 분류 등에 활발히 사용하고 있음을 확인하였다. 하지만 대부분의 연구에서 패킷데이터를 딥러닝 시키기 위한 데이터 변환 전처리과정이 포함되어 있음을 확인하였고 본 연구에서는 이 과정을 최소화하는 방향으로 탐지모델을 설계하였다.

3. 이미지 딥러닝 기반 DNS DDoS 탐지시스템

3.1 데이터셋

데이터셋은 L.F. Haaijer가 DDoS 연구목적으로 제공[17] 하는 18개의 “DDoS Packet Capture Collection”중 DNS 데이터셋은 표 1과 같이 DRDOS pcap 파일을 DRDoS 패킷

(표 1) 데이터셋 구성
(Table 1) structure of benign and attack dataset

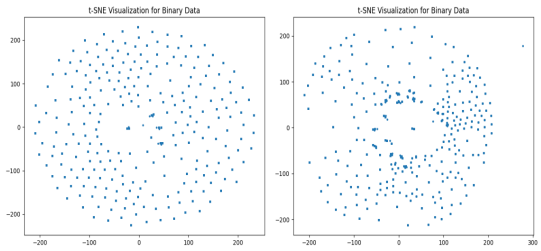
	attack	**benign**
총패킷수	12,000	54,863
평균패킷크기(byte)	512	122
udp구성비	udp 74.9%	udp 98.4%

(12,000개)으로 분류하고 정상적인 DNS 서비스 패킷은 국내 모 대학 소형 라우터에서 발생하는 일반적인 DNS 프로토콜 패킷(54,863개)만을 모아서 구성하였다.

3.2 패킷 이미지 생성

앞서 관련 연구에서 살펴보았듯이 대부분의 딥러닝에서 악성코드 시각화 및 패킷 시각화 메타 데이타를 추출하여 딥러닝에 적당한 데이터로 전처리하는 과정이 필요하다. 하지만 본 연구에서는 악성코드를 딥러닝으로 학습시켜 분류할 때 악성코드 바이너리 자체를 이미지화하여 학습시키는 방식을 응용[18] 하여 패킷을 전처리 과정 없이 이미지화하여 딥러닝으로 학습시켜 유의미한 탐지가 가능한지 연구해보았다.

우선 데이터 시각화 기법인 t-SNE(t-distributed Stochastic Neighbor Embedding)을 사용하여 데이터셋의 클러스터링과 거리, 분포 등을 살펴보았다.

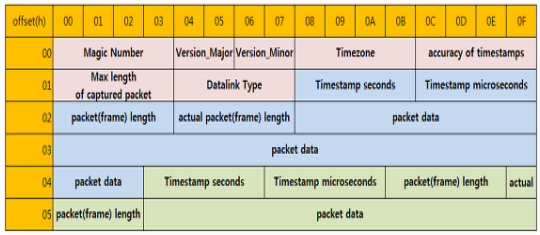


(그림 2) t-SNE 시각화 결과(L: DRDOS, R: 일반)
(Figure 2) t-SNE visulazation(L:attack R: normal)

그림 2처럼 일반 DNS 패킷파일(normal)은 명확한 군집(cluster)이 여러 개 보이고 각 군집은 서로 다른 패턴으로 구분이며 DNS DRDOS 패킷파일(attack)은 일반 DNS 패킷보다 넓게 분포되어 다양한 형태의 특징을 보이고 군집들이 거의 형성되어 있지 않은 독립적인 특징을 볼 수 있다.

이러한 시각화 분석을 통하여 두 패킷 데이터들의 비선형적인 유사성과 군집 특성 등 차이점을 파악할 수 있었으며 패킷 시각화를 기반으로한 딥러닝의 가능성을 파악하였다.

패킷캡쳐 파일인 PCAP 파일은 압축이 없는 무손실 파일형태로 그림 3처럼 패킷 헤더(16byte)와 패킷데이터가 반복되는 구조로 되어 있어 이를 그대로 잘라서 이미지화 시키면 패킷헤더가 잡음으로 작용하게 되므로 패킷에서 파일헤더와 패킷헤더를 제거하여 실제 패킷 데이터들의 연속성을 가질 수 있도록 하였다.



(그림 3) PCAP 파일 구조(패킷헤더 + 패킷 반복)
(Figure 3) PCAP File Structure

다음으로 수집한 패킷을 가지고 CNN을 이용하여 분류하기 위해서는 패킷 데이터를 이미지로 특징을 정형화해야 한다. 패킷의 경우 크기가 모두 다르므로 크기가 커지면 그만큼 이미지의 세로길이가 커진다. 이미지는 CNN 학습에 알맞도록 정사각형 모양의 이미지로 변환시키고 트래픽을 시각화할 경우 한 픽셀은 1바이트를 사용하기 때문에 0~ 255 범위로 256단계의 gray scale을 표현할 수 있다.

SY Lee 등[19]은 CNN으로 악성코드 파일 분류를 위해 악성코드 파일을 256 x 256 (65,536 byte) 사이즈의 이미지로 변환하였지만 큰 이미지 파일의 경우, 딥러닝 학습 시간이 이미지 크기에 따라 지수적으로 증가하고 실험 결과, 생성된 이미지는 작은 크기에도 시각적인 특징을 충분히 추출할 수 있으므로 패킷파일을 가로세로 길이 64 x 64 크기의 이미지로 저장하여 실험에 사용하였다.

(그림 4) 64 x 64 DRDOS 패킷 이미지
(Figure 4) 64 x 64 DNS DRDOS packet image

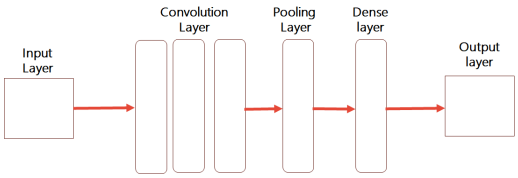
(그림 5) 64 x 64 일반 DNS 패킷 이미지
(Figure 5) 64 x 64 normal DNS packet image

그림 4, 5처럼 이미지화된 패킷을 보면 일반적인 정상 DNS 패킷들은 일반 사용자들의 비정형적인 특성을 반영한 이미지 패턴을 띄고 있으며 DNS DRDoS의 경우

DDoS 공격의 특성으로 인해 구간구간 반복적이고 규칙적인 패턴을 일부 보인다.

3.3 딥러닝 모델선정

앞서 관련 연구에서는 대부분의 트래픽 이미지에 대한 딥러닝 분석 논문에서 영상처리에 우수한 성능을 보이는 CNN을 주로 사용하였고 시간적인 특성을 추가하기 위한 RNN 계열의 딥러닝과 비교, LTSM 모델 등 다른 딥러닝 기법들과 결합하는 형태의 모델도 다수 연구되고 있음을 알 수 있었다. 이에 동 연구에서도 CNN을 기본 딥러닝 모델로 선정하고 실험하였다.



(그림 6) CNN 모델 구조
(Figure 6) CNN Model structure

CNN에서는 그림 6에서 보듯이 주로 Convolution Layer와 Pooling Layer를 사용하는데 Convolution layer는 각 이미지에서 특징을 추출하고 Pooling layer는 주요 특징이 아닌 부분은 삭제하는 형태로 이미지를 단순화 시킨다.

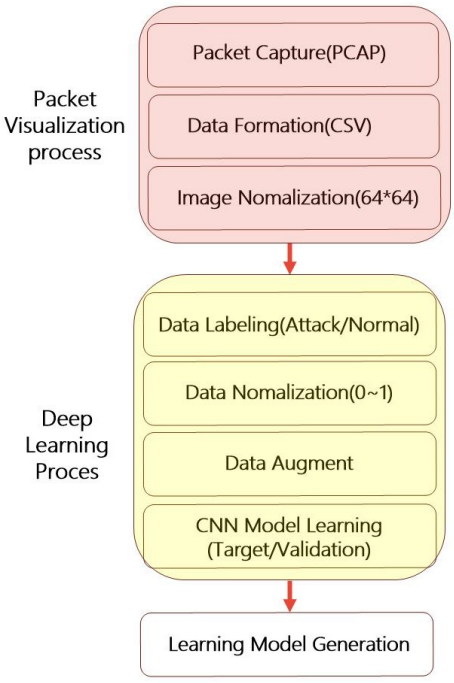
Convolution Layer가 많아질수록 모델은 더욱 정교해지지만, 학습 시간은 증가하게 된다. 전-연결계층(Fully Connected Layer)인 Dense Layer는 1차원만 학습할 수 있기 때문에 Flatten를 통하여 다차원을 1차원으로 만든 후 Dense Layer를 거쳐 중요한 특징만 남은 데이터를 추출하게 된다.

3.4 시스템 구성

이 장에서는 DNS DRDoS 공격을 탐지하기 위해 설계한 탐지시스템의 구성을 상세히 설명한다. 전체적인 구성도는 그림 7과 같다.

3.4.1 패킷 영상처리 및 정규화

패킷을 수집하여 적당한 크기로 잘라 흑백 이미지화한 후 Attack과 Benign데이터셋으로 구성하고 각 이미지는 정규화(Normalization)하는데 이를 통하여 픽셀 값 [0 ~ 255]을 [0, 1] 범위로 변환한다.



(그림 7) DNS DRDOS 탐지시스템 구성도
(Figure 7) DNS DRDOS Detection Model structure

이는 0에서 255 사이의 큰 값보다 수치적으로 더 안정적이며 활성화 함수인 Sigmoid 함수는 [0, 1] 범위의 입출력 값에 최적화되어 있으며 신경망에 비선형성을 추가해주는 활성화함수인 ReLU((Rectified Linear Unit) 함수도 정규화된 입력값에서 더 잘 작동한다.

3.4.2 데이터증강

이후 데이터 증강(Data Augmentation)을 통해 원본 데이터를 다양한 방식으로 변형하거나 확장하는데 데이터 부족 해결, 과적합(Overfitting) 방지와 모델의 일반화 능력 향상이 가능해진다. 이미지 데이터 증강 방법으로는 회전(Rotation), 좌우/상하 반전(Flip), 잘라내기(Cropping), 크기 조정(Scaling), 이동(Shift) 등이 사용된다.

3.4.3 CNN 분류기

본 논문에서 사용한 CNN 모델은 2개의 Convolution layer와 두 개의 Dense layer로 구성하게 된다. 이 Convolution layer을 통하여 이미지 전반의 특징을 추출하고 Max Pooling layer를 추가하여 중요한 특징만 추출하

는 방식으로 차원 축소 (Dimensionality reduction), 과적합 (Overfitting) 방지, 변환 불변성 (Translation invariance) 등의 이점을 얻는다.

한편, Convolution layer에서 활성화함수는 ReLu를 사용하는데, 이를 사용하면 비선형성을 부여하여 복잡한 패턴을 학습하게 하고 기존의 시그모이드 함수 출력 값이 0과 1 사이로 제한되고 입력 값이 커질수록 기울기가 0에 가까워지는 기울기 소실 문제(Vanishing Gradient Problem)를 완화시킨다.

64x64 크기의 이미지를 첫 번째 Convolution layer에 32개의 3x3 kernel로 필터링하여 32개의 64x64 이미지를 생성한다. 생성한 이미지에 ReLu를 적용한 후 2x2 Max Pooling layer을 거쳐 32개의 32x32 이미지를 생성한다.

두 번째 Convolution laye에서는 32개의 32x32 이미지를 입력으로 받아 64개의 3x3 kernel로 필터링하여 64개의 32x32이미지를 생성한다. 생성한 이미지에 ReLu를 적용한 후 2x2 Max Pooling layer를 거쳐 64개의 16x16 이미지를 생성한다.

생성된 이미지를 하나의 특징 데이터로 만든 16,384개의 값을 Flatten layer를 거쳐 3차원 데이터를 1차원 데이터로 만든 후 Dense layer를 통해 512개의 값을 출력한다.

이후 과적합을 방지하기 위해 일부 뉴런을 무작위로 비활성화하는 Drop-out 정규화를 거친 후 이 모델은 이진 분류(Binary Classification)모델 이므로 마지막 출력층 Dense layer의 출력 뉴런 개수는 1개이고 그 뉴런은 sigmoid 활성화 함수를 적용받아 0과 1 사이의 확률을 출력한다. (출력값이 0.5 이상이면 모델은 클래스 1로, 0.5 미만이면 클래스 0으로 분류) Optimizer는 Adam을 사용하였다.

4. 실 험

4.1 실험환경 및 데이터셋

탐지시스템은 Python으로 구현하였으며 Colab 환경에서 딥러닝 모델은 Tensorflow를 기반으로 작성하였다.

표 1의 데이터셋을 이미지화한 총 3,163개의 DNS DRDOS 이미지화 데이터와 일반 DNS 이미지화 데이터로 실험하였으며 표2처럼 70%를 학습(Training)에 사용하고 30%는 검증(Validation)에 사용하였다. 데이터셋을 학습 데이터와 검증 데이터로 구분하여 딥러닝을 학습시키고 탐지 정확도를 도출하였다.

(표 2) 정상 데이터와 이상 데이터 개수
(Table 2) Amount of benign and attack data

Training		Validation	
attack	benign	attack	benign
1020	1092	507	544

4.2 성능평가지표

4.2.1 혼동행렬

딥러닝 모델의 성능을 평가하기 위해 혼동행렬을 사용하는데 이는 분류 모델의 성능을 평가하는 데 사용하는 도구로, 예측 범주와 실제 데이터를 비교하여 시각적으로 나타낸 표로 모델이 얼마나 정확하게 분류했는지 성능을 평가할 때 사용한다.

표 3은 혼동행렬로 모델에서의 예측이 일반 DNS 트래픽이고, 실제 DNS 트래픽이 일반 DNS 트래픽일 때를 True Negative(TN), 모델은 DNS 공격 트래픽이라 판단했지만, 실제 트래픽은 일반 DNS 트래픽일 경우 False Positive(FP), 모델은 일반 DNS 트래픽이라 예측했지만 실제로는 공격일 경우는 False Negative(FN), 마지막으로 모델에서는 DNS 공격트래픽, 실제 파일도 DNS 공격 트래픽일 경우를 True Positive(TP)라 표현한다.

(표 3) 혼동행렬
(Table 3) Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

해당 4가지 TN, FP, FN, TP를 활용하여 정확도, 정밀도, 재현율, F1 점수 등을 계산할 수 있다.

· Accuracy = (TP + TN) / (TP +TN + FP + FN)

정확도(Accuracy)는 전체 데이터 중 정확하게 예측한 데이터의 수로 입력된 데이터를 얼마나 정확하게 예측하는가에 대한 비율이다. 정확도는 불균형 데이터셋에서는 왜곡된 결과를 보이기도 하고 거짓 긍정(False Positives) 또는 거짓 부정(False Negatives)에 민감도가 떨어진다. 이에 다음과 같은 보조 지표를 사용하게 된다.

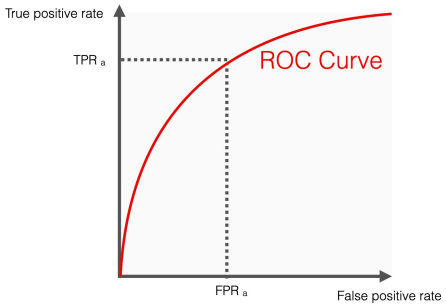
· Precision= TP / (TP + FP)
정밀도(Precision)는 양성(Positive)으로 예측한 데이터 중 실제로 양성에 속하는 샘플의 비율이다.

· Recall = TP / (TP + FN)
재현율(Recall)은 실제 양성(Positive)인 데이터 중에서 모델이 양성으로 잘 예측한 비율이다.
정밀도와 재현율은 상호의존적인 지표로 임의로 한쪽의 수치를 올리면 반대쪽은 떨어질 수 있다.
이에 정밀도와 재현율의 평균인 F1 Score를 사용하여 균형 잡힌 평가를 수행할 수 있다.

· F1 Score = 2 × (Precision × Recall) / (Precision + Recall)
정밀도와 재현율을 결합하여 만든 지표로 정밀도와 재현율의 조화 평균식이다. 불균형한 클래스에서 정밀도와 재현율 간의 정확한 평가를 위해 사용한다.

4.2.2 수신자 조작 특성곡선(RoC curve)

그림 8의 ROC(Receiver Operating Characteristic) curve는 이진분류에서 성능평가 방법으로 자주 사용되고 분류의 좋은 모델의 경우 좌측 모서리에 가까운 곡선의 형태를 그리게 된다. 이를 정량적으로 평가하는 방법에는 AUC(Area Under Curve)가 있는데 이는 RoC 커브의 아래면적을 나타내며 AUROC로 부르며 1을 최댓값으로 가진다.



(그림 8) RoC 커브
(Figure 8) RoC Curve

4.3 실험결과

최적의 CNN 모델을 구현하기 위해 Convolution Layer 별로 커널갯수를 조정하여 최적의 커널갯수를 찾아보고 Dense Layer output 크기와 Learning rate, dropout, Epoc값 등을 바꾸어가며 CNN 모델을 실험하여 성능을 비교해보았다. 마지막으로 이미지 분류에 사용되는 기계학습들과의 성능 비교를 통하여 제안한 모델의 성능을 평가해 보았다.

4.3.1 CNN Convolution Layer와 커널 갯수 조정

Convolution layer를 몇 계층으로 할 것인가 결정하기 위해 우선 3계층로 실험해본 결과 0.998 이상의 정확도를 보였으나 2계층으로 실험해본 결과도 0.996 ~ 0.998 정도의 정확도를 나타내었다.

Convolution layer가 늘어날수록 성능은 우수하나 학습시간이 많이 소요된다는 단점이 있다. 따라서 본 연구에서는 Convolution layer를 2개 사용하기로 결정하였고 각 Convolution layer 별로 커널 갯수를 표 4처럼 16, 32, 64, 128 사이즈로 바꿔가며 Convolution layer별 최적의 커널 갯수를 찾아보았다. 최고의 성능보다는 우수한 성능을 가지고도 과적합 등이 발생하지 않는 안정된 모델을 찾는 것을 목표로 하였다. (epoc=10, Learning rate = 0.001, dropout = 0.5, Dense Layer 출력값 128으로 고정)

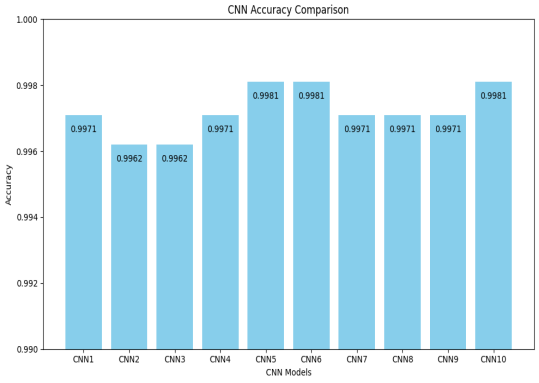
(표 4) Convolution layer 커널갯수 변화에 다른 정확도
(Table 4) Accuracy by Number of Kernels

CNN	CL1	CL2	정확도
CNN1	16	16	0.9971
CNN2	16	32	0.9962
CNN3	16	64	0.9962
CNN4	16	128	0.9971
CNN5	32	32	0.9981
CNN6	32	64	0.9981
CNN7	32	128	0.9971
CNN8	64	64	0.9971
CNN9	64	128	0.9971
CNN10	128	128	0.9981

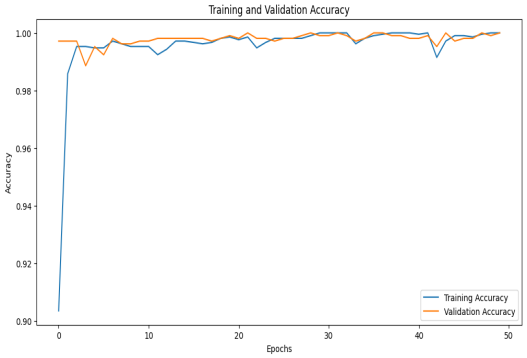
여러 조합으로 실험해 본 결과 커널 갯수가 많을수록 성능이 좋아지지만, 표 4의 CNN7, CNN8, CNN9 모델의 경우 큰 커널 갯수의 조합임에도 작은 커널 갯수를 가진 CNN5, CNN6보다 성능이 일부 떨어지는 현상도 나타나서 커널 개수가 많다고 하여 항상 성능이 우수하다고 할 수 없다.

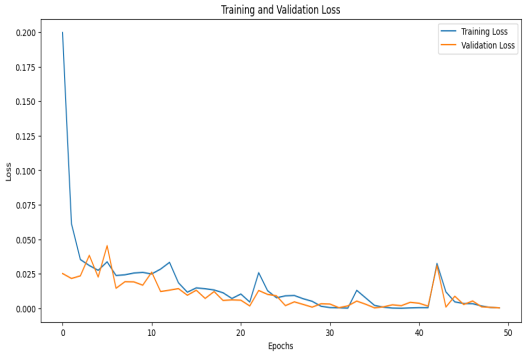
본 연구에서는 적절한 커널 갯수를 찾기 위해 정확도 0.9981로 가장 우수한 성능을 보인 CNN5, CNN6, CNN10 모

델 중에서 Layer 별로 다른 커널 갯수를 사용하고 그림 10, 11와 같이 훈련 및 검증 정확도 그래프가 타 모델보다 안정적인 CNN6 모델(32, 64)을 주 실험 모델로 선정하였다.

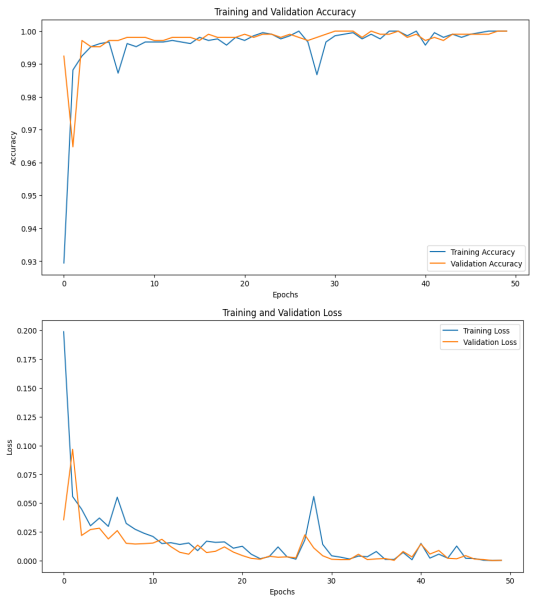


(그림 9) CNN모델별 정확도
(Figure 9) CNN comparison by accuracy





(그림 10) Convolution layer 32 x 64 성능
(Figure 10) Convolution layer 32 x 64 graph



(그림 11) Convolution layer 64 x 128 성능
(Figure 11) Convolution layer 64 x 128 graph

4.3.2 Dense Layer 출력값 조정

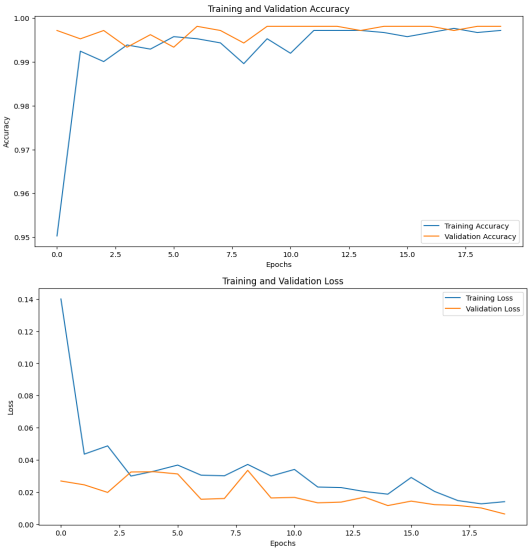
CNN6 모델에서 가장 적절한 Dense Layer 출력값(output)을 찾기 위해 표 5와 같이 16 ~ 4096까지 두 배씩 증가시켜 조정하면서 정확도를 비교해 보았다. (20 epoch, Learning rate = 0.001, dropout = 0.5로 고정)

16부터 4,096까지 변화시켜본 결과, 정확도는 0.997 ~ 0.999 사이를 보였으며 실행시간은 출력값이 두 배로 증가할 때 로그적으로 증가한다. 실험을 통해 성능을 비교하여 가장 작은 값으로 효과적인 성능을 나타내는 값을 찾아보았다.

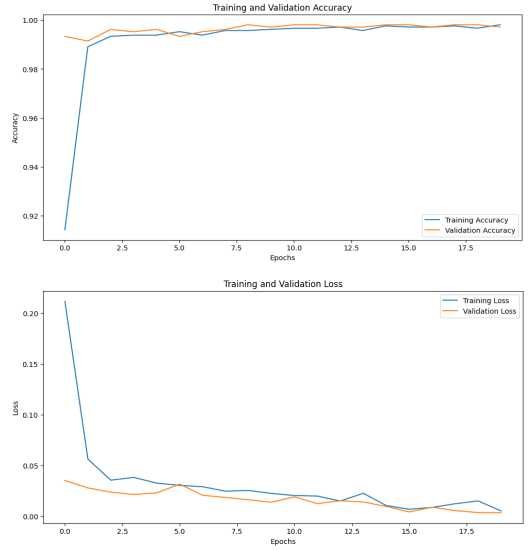
(표 5) Dense Layer 출력값에 따른 비교
(Table 5) Comparison of Dense Layer output

DL output	Accuracy	Precision	recall	F1 score
16	0.9971	0.9972	0.9971	0.9971
32	0.9981	0.9982	0.9980	0.9981
64	0.9971	0.9973	0.9970	0.9971
128	0.9981	0.9982	0.9980	0.9981
256	0.9943	0.9942	0.9944	0.9943
512	0.9990	0.9991	0.9990	0.9990
1024	0.9981	0.9982	0.9980	0.9981
2048	0.9990	0.9990	0.9991	0.9990
4096	0.9971	0.9971	0.9972	0.9971

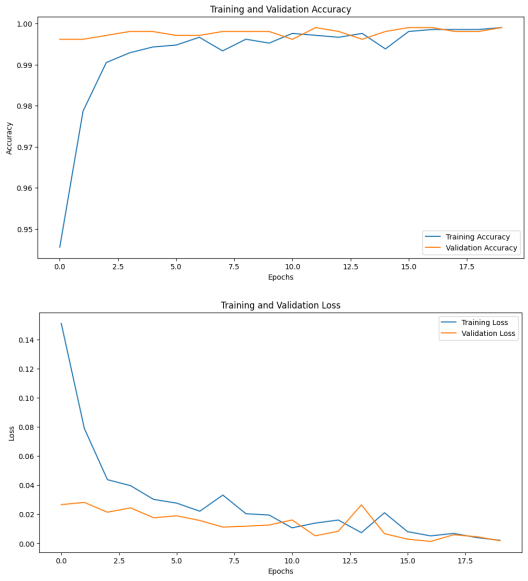
256을 제외하고는 모두 0.997 이상의 정확도를 보이고 있어, 모델의 훈련 정확도와 검증 정확도 그래프를 통하여 비교해 보았다. 정확도가 0.998 이상인 Dense layer out을 가진 그림 12, 13, 14를 비교해 볼 때 출력이 512일 때 학습 동안 안정적인 훈련 및 검증 정확도를 보였다.



(그림 12) Dense Layer 출력값 128의 성능
(Figure 12) Dense Layer output 128 graph



(그림 13) Dense Layer 출력값 512 성능
(Figure 13) Dense Layer output 512 graph



(그림 14) Dense Layer 출력값 2048
(Figure 14) Dense Layer output 2048

4.3.3 학습률(Learning Rate) 조정

학습률(Learning rate)이 크면 속도는 빠르지만, Training 과정에서 발생하는 오류를 줄이지 못하고 과적합이 생기고 학습률이 너무 낮은 경우에는 학습 과정이 오래 걸리고 Validation시 오류 값이 너무 많아진다. CNN6 모델에서 적절한 학습률을 찾기 위해 표6처럼 0.1 ~ 0.00001까지 학습률을 바꾸어가며 성능을 측정해 보았다. (30 epoch, Dense layer output 256, dropout = 0.5로 고정) 실험 결과, 학습률 0.1은 학습하지 못하였으며 0.01도 성능은 좋았으나 가끔 발산하는 경향을 보였다. 0.001이 가장 우수한 성능을 보였고 0.0001과 0.00001은 안정성은 높았으나 오히려 성능이 떨어졌다. 이는 적은 Epoch 수로 인하여 높은 학습률이 성능이 나오지 않는 경우로 판단 된다.(4.3.6의 Learning rate와 Epoch 수의 상관관계 참고)

(표 6) 학습률 변화에 따른 비교
(Table 6) Comparison of Learning rate

학습률	Accuracy	Precision	recall	F1 score
0.1	0.5176	0.2588	0.5000	0.3411
0.01	0.9981	0.9982	0.9980	0.9981
0.001	0.9990	0.9990	0.9991	0.9990
0.0001	0.9933	0.9936	0.9931	0.9933
0.00001	0.9943	0.9945	0.9941	0.9943

4.3.4 dropout 조정

딥러닝 모델에서는 과적합을 방지하기 위한 정규화 기법의 하나로 Dropout 기능을 사용하는데 이 방법은 학습 과정에서 무작위로 일부 뉴런을 비활성화하여 모델이 특정 뉴런에 과도하게 의존하지 않게 하는 일반화 기법이다. Dropout 0.1이라면 학습 중 뉴런의 10%만 무작위로 비활성화한다. 비활성화 비율이 적어 일반화 성능은 떨어진다. 여기서는 표 7과 같이 CNN6 모델에서 0.1 ~ 0.9까지 dropout을 바꾸어가며 최적의 값을 찾아보았다.

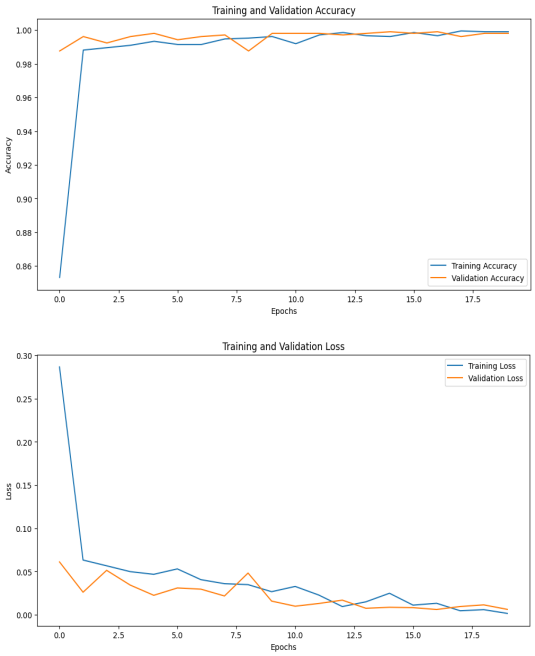
(표 7) dropout 변화에 따른 비교
(Table 7) Comparison of dropout

DO	Accuracy	Precision	recall	F1 score
0.0	0.9971	0.9972	0.9972	0.9971
0.1	0.9981	0.9982	0.9980	0.9981
0.2	0.9990	0.9991	0.9990	0.9990
0.3	0.9990	0.9991	0.9990	0.9990
0.4	0.9981	0.9982	0.9980	0.9981
0.5	0.9981	0.9982	0.9980	0.9981
0.6	0.9981	0.9982	0.9980	0.9981
0.7	0.9971	0.9972	0.9971	0.9971
0.8	0.9971	0.9973	0.9970	0.9971
0.9	0.9971	0.9973	0.9970	0.9971

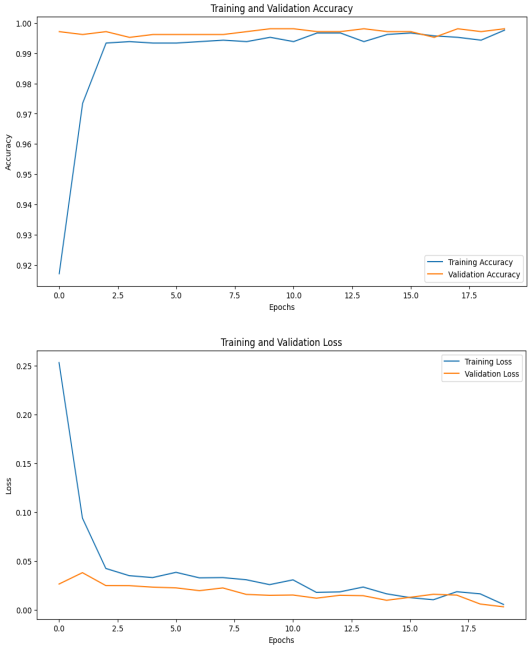
실험 결과(20 epoch, Dense layer output 256, learning rate = 0.001로 고정) , dropout이 0.2, 0.3일 때 정확도가 가장 우수하였으나 모든 성능지표가 동일하여 그림 15, 16과 같이 훈련 및 검증 정확도 그래프를 비교한 결과, 0.3일 때 보다 학습과정이 안정적임을 확인할 수 있었다.

4.3.5 배치(Batch)크기 변화에 따른 성능

딥러닝 모델에서 배치 크기는 모델을 훈련할 때 한 번에 처리하는 샘플의 수를 의미하는데 작은 배치 크기는 훈련 데이터의 더 많은 변화를 반영할 수 있어 모델의 일반화 성능을 향상시키는 데 도움이 되고 큰 배치 크기는 안정적인 경향을 보여서 손실 함수의 최소값을 더 잘 찾을 수 있다. 표 8과 같이 배치파일의 크기를 8 ~ 256까지 바꾸어가며 실험한 결과 동일한 성능으로 측정된 경우에도 그림 17, 18처럼 배치파일의 크기가 클수록 훈련 및 검증 정확도가 안정이었으며 실행속도가 빨랐다. CNN6 모델에서는 정확도가 가장 높은 32를 선정하였다. (30 epoch, Dense layer output 256, learning rate = 0.001, dropout 0.5로 고정)



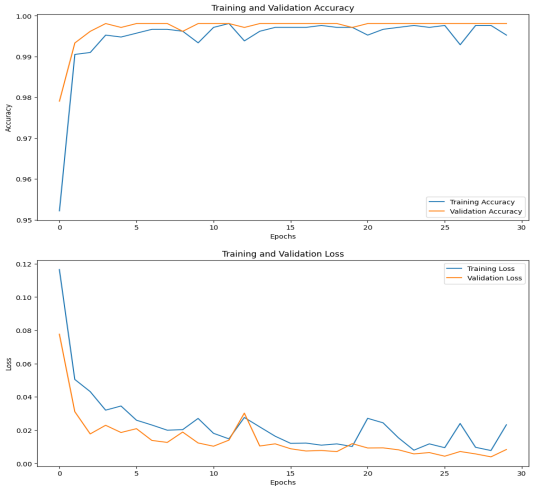
(그림 15) 드롭아웃 0.2 일 때 학습 성능
(Figure 15) Learning late at Dropout 0.2



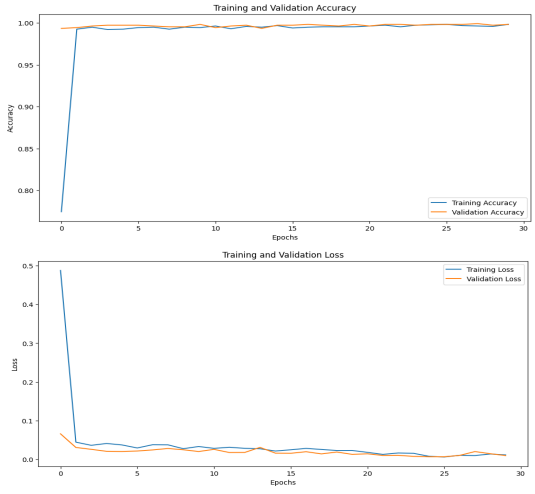
(그림 16) 드롭아웃 0.3 일 때 학습 성능
(Figure 16) Learning late at Dropout 0.3

(표 8) 배치크기 변화에 따른 비교
(Table 8) Comparison of batch size

배치크기	Accuracy	Precision	recall	F1 score
8	0.9981	0.9982	0.9980	0.9981
16	0.9971	0.9973	0.9970	0.9971
32	0.9990	0.9990	0.9991	0.9990
64	0.9962	0.9962	0.9962	0.9962
128	0.9981	0.9982	0.9980	0.9981
256	0.9981	0.9982	0.9980	0.9981



(그림 17) 배치크기 8 학습 성능
(Figure 17) Learning late at Batch size 8



(그림 18) 배치크기 128 학습 성능
(Figure 18) Learning late at Batch size 128

4.3.6 Epoch 변화에 따른 성능

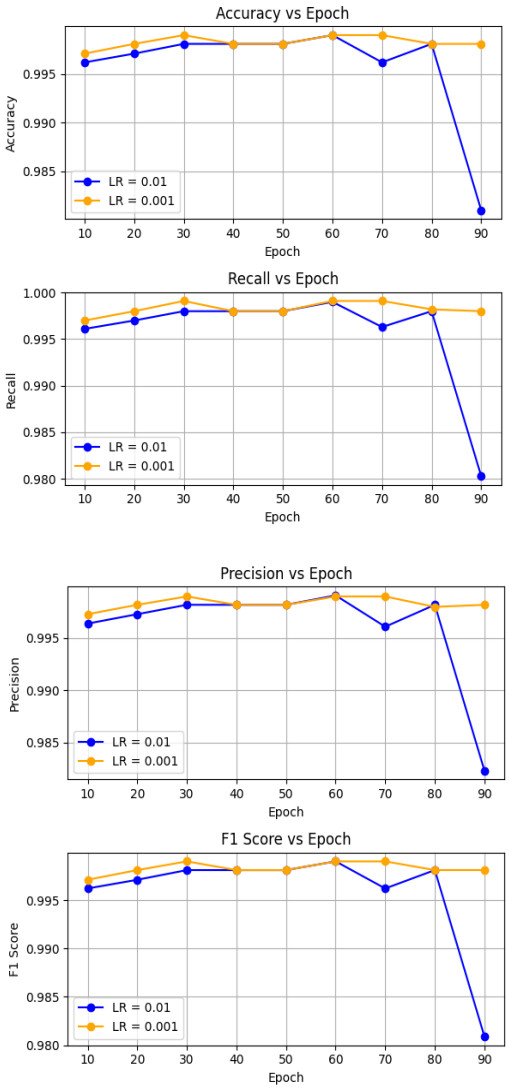
딥러닝에서 Epoch는 전체 훈련 데이터셋이 한 번 모델을 통해 전달되어 학습하는 과정을 의미한다. 각 Epoch 동안 모델은 훈련 데이터를 보고 가중치를 업데이트하며 여러 Epoch 동안 학습함으로써 모델은 데이터의 패턴을 점차적으로 상세하게 학습하게 된다. Epoch 수가 너무 많으면 모델이 훈련 데이터에 과적합되어 새로운 데이터에 대한 일반화 성능이 떨어질 수 있다. 매개변수들은 고정(Dense layer output 256, learning rate = 0.01/0.001, dropout = 0.5로 고정) 하고 표9, 10와 같이 10 ~ 90까지 Epoch 값의 변화에 따른 동 모델의 성능을 측정해보았다.

(표 9) Epoch 변화에 따른 비교(LR=0.01)
(Table 9) Comparison of Epoch(LR=0.01)

Epoc	Accuracy	Precision	recall	F1 score
10	0.9962	0.9964	0.9961	0.9962
20	0.9971	0.9973	0.9970	0.9971
30	0.9981	0.9982	0.9980	0.9981
40	0.9981	0.9982	0.9980	0.9981
50	0.9981	0.9982	0.9980	0.9981
60	0.9990	0.9991	0.9990	0.9990
70	0.9962	0.9961	0.9963	0.9962
80	0.9981	0.9982	0.9980	0.9981
90	0.9810	0.9823	0.9803	0.9809

(표 10) Epoch 변화에 따른 비교(LR=0.001)
(Table 10) Comparison of Epoch(LR=0.001)

Epoch	Accuracy	Precision	recall	F1 score
10	0.9971	0.9973	0.9970	0.9971
20	0.9981	0.9982	0.9980	0.9981
30	0.9990	0.9990	0.9991	0.9990
40	0.9981	0.9982	0.9980	0.9981
50	0.9981	0.9982	0.9980	0.9981
60	0.9990	0.9990	0.9991	0.9990
70	0.9990	0.9990	0.9991	0.9990
80	0.9981	0.9980	0.9982	0.9981
90	0.9981	0.9982	0.9980	0.9981



(그림 19) Epoch값에 따른 성능비교(LR=0.01/0.001)
(Figure 19) Performace Comparison by Epoch Value (LR=0.01/0.001)

그림 19를 보면 Learning rate가 0.01인 경우 초기에는 일부 좋은 성능을 보이나 Epoch가 늘어나면서 학습이 불안정하여 성능이 오히려 떨어지기도 하는데 표 10처럼 Learning rate가 0.001인 경우, Epoch 40 이후부터는 Learning rate가 0.01보다 성능과 안정성 모두 앞서게 된다. 이는 Learning rate 값이 낮은 경우, Epoch가 증가할수록 서서히 학습 성능이 증가하는 과정으로 볼 수 있다.

4.3.7 제안한 CNN6 모델 성능평가

상기 여러 실험을 통해 제안한 CNN6 모델에 적합한 초매개변수들은 표 11과 같이 정리할 수 있었다.

(표 11) CNN6 모델의 초매개변수
(Table 11) Hyper parameters of CNN 6 model

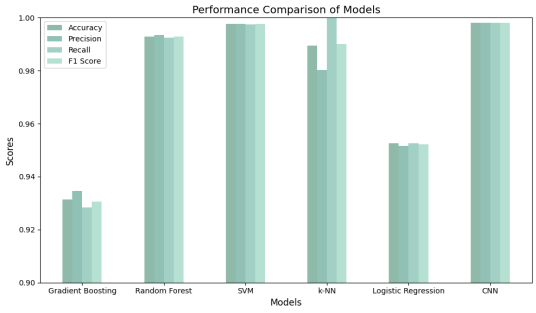
Model Parameter Setting	
Input vector	64 x 64
Convolution Layer	2
커널의 개수	32, 64
커널 크기	3 x 3
Pooling Layer	2 x 2
Activation fuction	Relu, sigmoid
Batch size	32
Epoch	100
Dropout	0.3
Dense Layer	512
Optimizer	Adam
Learning rate	0.001

상기 매개변수들을 적용한 CNN 실험결과 모델의 정확도는 0.998, 손실도는 0.0171이었다. 한편, 이미지 분류는 기계학습으로도 가능하므로 일반적인 기계학습모델 중에서 표12과 같이 SVM, Random Forest, k-NN, logistic Regulation, Gradient Boosting 모델 등과 성능을 비교해 보았다.

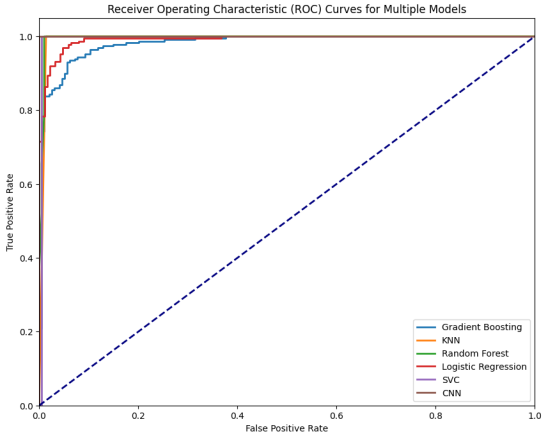
DecisionTree, Naive Bayes도 기계학습 모델이지만 이미지의 특성상 각 픽셀을 독립적인 특성으로 간주하고 이를 기준으로 데이터를 분할하게 되는데 이미지의 경우 인접 픽셀 간의 연관성(예: 모양, 패턴, 경계)을 학습하는 것이 중요하기에 여기서는 비교모델에서 제외하였다.

(표 12) 기계학습들과 제안한 CNN 모델 성능 비교
(Table 12) Comparison with Machine Learning

	Accuracy	Precision	recall	F1 score
Gradient Boosting	0.9314	0.9345	0.9284	0.9305
Radom Forest	0.9929	0.9934	0.9925	0.9929
SVM	0.9976	0.9977	0.9975	0.9976
KNN	0.9895	0.9802	1.0000	0.9900
Logistic Regulation	0.9527	0.9517	0.9527	0.9522
CNN (e=30)	0.9981	0.9981	0.9981	0.9981



(그림 20) 기계학습들과 CNN 성능 비교표
(Figure 20) CNN Comparison with ML



(그림 21) 기계학습들과 CNN 성능 비교(ROC)
(Figure 21) CNN Comparison with ML(ROC)

그림 20을 보면 기계학습 중에서는 SVM이 가장 우수한 성능을 보였으며 여타 기계학습을 이용한 이미지 분류 방식도 0.95정도의 정확도를 보였지만 기계학습 결과, 64x64 사이즈로 줄인 이미지에서는 주요 특징을 추출하기가 딥러닝 모델인 CNN6에 비해 성능이 떨어졌으며 KNN을 제외하고는 실행시 마다 성능이 변하여 일관성 있고 안정적인 성능을 도출하지 못했다. 표12에서 CNN6 정확도는 0.9981로 딥러닝의 특성상 실행시 마다 약간 변하였지만 대부분 0.995이상이었다.

한편, 제안한 CNN6 모델은 그림 20, 21에서 보듯이 기계학습에 비해 반복 실험시에도 성능지표가 안정적이고 우수한 성능을 보였으며 이미지 해상도가 낮은 경우에도 CNN을 이용한 딥러닝에 최적의 파라미터 조합을 적용하는 경우 충분히 좋은 성능을 나타낼 수 있음을 확인할 수 있었다.

5. 결 론

DRDoS에 대한 기계학습과 딥러닝을 이용한 탐지 방법들이 연구되어 왔으나, 대부분의 기존 방법은 패킷의 특징값을 추출하여 학습시키는 방식이기 때문에 실시간 탐지가 요구되는 상황에서 데이터 전처리 과정에 많은 시간이 소요되는 문제가 있다.

본 논문에서는 이러한 한계를 극복하기 위해 악성코드 전체 실행 코드를 이미지화하여 학습시키는 방법에서 착안하여, DNS DRDoS 공격 패킷들을 이미지화하여 CNN을 이용한 학습 방법으로 해당 공격을 탐지하는 접근을 시도하였다. 실험 결과, 제안된 방법은 일반적인 기계학습모델에 비해 높은 탐지율(99.80%)을 유지하면서도 데이터 전처리 과정을 생략함으로써 신속한 탐지가 가능하다는 장점을 보여주었다.

DDoS의 특성상 100%에 가까운 정확한 탐지보다는 실시간 대응이 중요한 현장에서는 탐지율보다 빠른 탐지가 더욱 중요한 요소가 될 수 있다. 본 연구는 실시간 네트워크 환경에서 DNS DRDoS 공격을 조기에 탐지하고 대응할 수 있는 효과적인 솔루션을 제안하였다.

향후 연구에서는 다른 유형의 DDoS 공격에도 이미지 기반 학습 기법을 적용하여 탐지 및 공격 유형의 분류 가능성을 탐구하고 다양한 이미지 크기의 데이터셋과 여러 출처의 DDoS 데이터셋을 활용하여 탐지모델의 일반화 성능을 향상시키는 방향으로 연구를 확대할 예정이다. 또한, CNN에 국한하지 않고 PCA와 LBP 등 이미지 특징 추출기법과 딥러닝을 결합한 탐지모델, 그리고 CNN과 LSTM 등 타 딥러닝 기법의 결합한 탐지모델 등 실시간 시스템에 최적화되고 보다 경량화되고 빠른 성능을 보이는 딥러닝 기반 탐지모델을 구현할 예정이다.

참고문헌(Reference)

[1] https://www.isssource.com/huge-ddos-attack-a-new-approach/
[2] http://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/
[3] https://www.eweek.com/security/github-hit-by-largest-ddos-attack-ever-recorded-at-1.35-tbps
[4] https://blog.cloudflare.com/ddos-threat-report-for-2024-q1-ko-kr
[5] HJ Kim, KH Han, SS Shin, “DRDoS amplication attack response system,” Journal of Convergence for

Information Technology, Vol. 10, No. 12, pp. 22-30, 2022. https://doi.org/10.22156/CS4SMB.2020.10.12.022

[6] H Aydın, “A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment,” Computers & Security, Vol. 118, pp.102725, 2022. https://doi.org/10.1016/j.cose.2022.102725

[7] SY Diaba, M Elmusrati, “Proposed algorithm for smart grid DDoS detection based on deep learning,” Neural Networks, Vol. 159, pp. 175-184, 2023. https://doi.org/10.1016/j.neunet.2022.12.011

[8] JD Gadze et al,, “An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers,” Technologies, 9(1), 14, 2021. https://doi.org/10.3390/technologies9010014

[9] A Singh and J Jang-Jaccard, Autoencoder-based Unsupervised Intrusion Detection using Multi-Scale Convolutional Recurrent Networks, 2022. https://doi.org/10.48550/arXiv.2204.03779

[10] FM Aswad et al., “Deep learning in distributed denial-of-service attacks detection method for Internet of Things networks,” Journal of Intelligent Systems, 32(1), 20220155, 2023. https://doi.org/10.1515/jisys-2022-0155

[11] TU Ahmed, et al., “An Integrated CNN-RNN Framework to Assess Road Crack,” 2019 22nd International Conference on Computer and Information Technology (ICCIT), pp. 1-6, 2019. https://doi.org/10.1109/ICCIT48885.2019.9038607

[12] In Hyuk Seo, Ki-Taek Lee, Jinhyun Yu, Seungjoo Kim, “CNN Based Real-Time DNS DDoS Attack Detection System,” KIPS Transactions on Computer and Communication Systems, Vol. 6, No. 3, pp. 135-142, Mar. 2017. http://dx.doi.org/10.3745/KTCCS.2017.6.3.135

[13] Kimoon Jeong, “A Study of Data Preprocessing for Network Intrusion Detection based on Deep Learning,” Proceedings of the 2018 Korea Computer Information Society (KCIS) Summer Conference. Vol. 26 No. 2, pp. 165-166, 2018. https://koreascience.kr/article/CFKO201831342440410.page

[14] Kim T, Pak W., “Deep Learning-Based Network Intrusion Detection Using Multiple Image Transformers,” Applied Sciences, 13(5), 2754, 2023. https://doi.org/10.3390/app13052754

[15] Kim J, Kim J, Kim H, Shim M, Choi E. “CNN-Based Network Intrusion Detection against Denial-of-Service Attacks,” Electronics, 9(6), 916, 2020. https://doi.org/10.3390/electronics9060916

[16] Y. He and W. Li, “Image-based Encrypted Traffic Classification with Convolution Neural Networks,” 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), pp. 271-278, 2020. http://dx.doi.org/10.1109/DSC50466.2020.00048

[17] L.F. Haaijer, DDoS Packet Capture Collection, 2022. https://github.com/StopDDoS/packet-captures

[18] Nataraj L, Karthikeyan S, Jacob G, Manjunath B. S., “Malware images: visualization and automatic classification,” in Proc. of the 8th ACM international symposium on visualization for cyber security, 2011. http://dx.doi.org/10.1145/2016904.2016908

[19] SY Lee, B Moon, J Kim, “Malware Classification Schemes Based on CNN Using Images and Metadata,” Proceedings of the Korea Information Processing Society Conference, 2021. https://doi.org/10.3745/PKIPS.y2021m05a.212

◐ 저 자 소 개 ◑

신 훈(Hoon Shin)
1992년 한국항공대학교 컴퓨터공학과(이학사)
1994년 서강대학교 컴퓨터공학과(공학석사)
1994년~1996년 KIDA 국망망보안팀
1996년~1998년 KISA CERTCC-KR
2024년~현재 세종대학교 대학원 컴퓨터공학과(박사과정)
관심분야 : 정보보호, AI보안, 네트워크보안, 머신러닝 etc.
E-mail : kadosu@hanmail.net

정 재 영(Jae-yeong Jeong)
2021년 숭실대학교 정보보안학과(학사)
2021년~2024년 세종대학교 대학원 컴퓨터공학과(공학석사)
2024년~현재 세종대학교 대학원 컴퓨터공학과(박사과정)
관심분야 : 네트워크, 데이터마이닝, 딥러닝, etc
E-mail : jaeyeong@sju.ac.kr

조 규 민(Kyu-min Cho)
1993년 서울대학교 계산통계학과(이학사)
2002년 동국대학교 정보보호대학원 정보보호학과(공학석사)
2015년~현재 금융보안원 부장
2023년 세종대학교 대학원 컴퓨터공학과(박사)
관심분야 : 정보보호, AI보안, 금융보안, 사이버레질리언스 etc.
E-mail : gmcho69@naver.com

이 재 일(Jae-il Lee)
1986년 서울대학교 계산통계학과(이학사)
1992년 서울대학교 계산통계학과(이학석사)
2006년 연세대학교 컴퓨터과학과(공학박사)
1996년~2023년 한국인터넷진흥원
2003년~현재 스마일게이트
관심분야 : 침해사고 대응, PKI, 인증, 모바일보안, etc.
E-mail : jilee0218@gmail.com

신 동 규(Dong-kyoo Shin)
1986년 서울대학교 계산통계학과(이학사)
1992년 Illinois Institute of Technology 대학원 컴퓨터과학과(공학석사)
1997년 Texas A&M University 대학원 컴퓨터과학과(공학박사)
1998년~현재 세종대학교 컴퓨터공학과 교수
관심분야 : 머신러닝, 유비쿼터스 컴퓨팅, 생체신호 데이터처리, 정보보호, etc.
E-mail : shindk@sejong.ac.kr