



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학전문석사 학위 연구보고서

네트워크 트래픽 이상징후
탐지율 향상을 위한 자기지도학습
기반의 오토인코더 최적화 연구

Self-supervised Network Anomaly Detection
using Autoencoders

2020 년 2 월

서울대학교 공학전문대학원

응용공학과 응용공학전공

서 승 수


네트워크 트래픽 이상징후
탐지율 향상을 위한 자기지도학습
기반의 오토인코더 최적화 연구


지도 교수 윤 성 로

이 프로젝트 리포트를 공학전문석사 학위
연구보고서로 제출함
2020 년 2 월

서울대학교 공학전문대학원
응용공학과 응용공학전공
서 승 수

서승수의 공학전문석사 학위 연구보고서를 인준함
2020 년 2 월

위 원 장 _____ 구 윤 모 

위 원 _____ 윤 성 로 

위 원 _____ 곽 우 영 

국문초록

네트워크 기술의 발전에 따라 네트워크 트래픽이 급격히 증가하고 있다. 이와 함께 네트워크를 통한 사이버 위협 역시 나날이 늘어가면서 다양한 형태의 네트워크 공격을 효율적으로 탐지할 수 있는 방법이 요구되고 있다. 네트워크 침입탐지시스템에서 주로 사용되고 있는 시그니처 탐지 방식은 이미 알려진 공격 유형에는 효과적이지만, 새로운 공격 유형에는 즉각적인 탐지 및 대처가 어렵다. 이상징후 탐지 방식은 새로운 공격 유형에 대한 탐지가 가능하지만, 오탐이 발생할 가능성이 높아 실제 보안 시스템에 적용하기에는 아직 부족한 실정이다.

최근 딥러닝 기술이 다양한 분야에서 우수한 성능을 보이면서, 네트워크 침입탐지시스템에도 딥러닝을 활용하는 연구가 활발히 이루어지고 있다. 하지만 대다수의 연구에서 사용되는 데이터셋은 수집 시기가 너무 오래되어 최신 네트워크 침입 패턴을 제대로 반영하지 못하고, 실제 네트워크 트래픽을 완벽하게 표현할 수 없다는 한계가 있다. 게다가 지도학습 기반의 대다수 탐지 모델은 학습에 사용할 데이터를 레이블링해야 하는 어려움이 있고, 네트워크 트래픽의 특성상 클래스 불균형에 따른 문제가 발생할 수 있다. 따라서 이러한 문제를 해결할 수 있는 보다 지능적이고 효과적인 네트워크 침입탐지 기법이 필요하다.

본 연구에서는 정상 트래픽 데이터만으로 모델을 학습할 수 있는 오토인코더를 이용한 네트워크 침입탐지 기법을 제안한다. 그리고 최근에 수집된 트래픽 데이터셋을 사용하여 모델의 탐지 성능을 확인한다. 제안 모델의 적합성을 판단하기 위해, 지도학습 기반의 서포트벡터머신 모델, 심층신경망 모델과 탐지 결과를 비교하고 그 효과를 검증한다. 실험을 통해 제안한 모델의 최적화된 구조를 찾고 그 성능을 검증한 결과, 탐지 성능이 서포트벡터머신 모델보다 F1 스코어 기준 5%p 이상 높게 측정되었고, 기존 두 모델보다 새로운 공격 유형에 대한 탐지율이 우수함을 확인했다. 본 연구에서 제안한 자기지도학습 기반의 오토인코더를 이용한 네트워크 이상징후 탐지 모델은 날이 갈수록 진화하는 사이버 위협에 대응할 수 있는 네트워크 침입탐지시스템의 기반이 될 것으로 기대한다.

주요어: 네트워크 침입탐지시스템, 이상징후 탐지, 딥러닝, 심층신경망, 오토인코더

학 번: 2018-21440

목 차

제 1 장 서 론	1
제 1 절 연구 배경	1
제 2 절 연구 내용 및 목적	2
제 3 절 연구 보고서 구성	3
제 2 장 배경 지식과 관련 연구	4
제 1 절 네트워크 침입탐지시스템	4
1. 시그니처 탐지	5
2. 이상징후 탐지	6
제 2 절 딥러닝 모델	6
1. 심층신경망	6
2. 오토인코더	10
제 3 절 딥러닝 기반 네트워크 이상징후 탐지 연구	12
제 3 장 연구 방법	15
제 1 절 네트워크 트래픽 데이터셋	15
제 2 절 네트워크 이상징후 탐지 모델	17
1. 서포트벡터머신 기반 탐지	17
2. 심층신경망 기반 탐지	17
3. 오토인코더 기반 탐지	19
제 4 장 실험 및 결과	21
제 1 절 실험 개요 및 환경	21
1. 실험 개요	21
2. 실험 환경	21
3. 성능 평가 지표	22
제 2 절 실험 내용 및 결과	24
1. 실험 데이터 분석 및 전처리	24
2. 서포트벡터머신 기반 네트워크 이상징후 탐지 모델	29
3. 심층신경망 기반 네트워크 이상징후 탐지 모델	31
4. 오토인코더 기반 네트워크 이상징후 탐지 모델	33
5. 성능 평가	39
제 5 장 결 론	42
제 1 절 연구 성과	42
제 2 절 보완 사항 및 향후 계획	43
참고문헌	44
Abstract	47

표 목차

[표 2 -1] 활성화 함수 종류	9
[표 3 -1] 네트워크 트래픽 데이터셋 비교	16
[표 3 -2] 서포트벡터머신 모델의 파라미터	17
[표 3 -3] 심층신경망 모델의 하이퍼 파라미터 설정	18
[표 3 -4] 오토인코더 모델의 하이퍼 파라미터 설정	20
[표 4 -1] 실험 환경 정보	22
[표 4 -2] 네트워크 트래픽 데이터의 오차행렬	22
[표 4 -3] CICIDS2017 트래픽 플로우 데이터 예시	24
[표 4 -4] CICIDS2017 특징 정보	25
[표 4 -5] CICIDS2017 주요 특징 정보	26
[표 4 -6] CICIDS2017 정상 트래픽 샘플 수	26
[표 4 -7] CICIDS2017 공격 트래픽 유형별 샘플 수	27
[표 4 -8] 최종 데이터의 유형별 샘플 수	29
[표 4 -9] 서포트벡터머신 모델의 탐지 성능	30
[표 4 -10] 서포트벡터머신 모델의 새로운 공격 탐지율	31
[표 4 -11] 심층신경망 모델의 탐지 성능	31
[표 4 -12] 심층신경망 모델의 새로운 공격 탐지율	33
[표 4 -13] 기본 오토인코더 모델의 탐지 성능	34
[표 4 -14] 잡음제거 오토인코더 모델의 탐지 성능	35
[표 4 -15] 희소 오토인코더 모델의 탐지 성능	35
[표 4 -16] 적층 오토인코더 모델의 탐지 성능	35
[표 4 -17] 적층 희소 오토인코더 모델의 탐지 성능	36
[표 4 -18] 오토인코더 모델별 탐지 성능	38
[표 4 -19] 오토인코더 모델별 새로운 공격 탐지율	39
[표 4 -20] 네트워크 이상징후 탐지 모델의 성능 비교	40

그림 목차

[그림 2 -1] SPAN 방식의 네트워크 침입탐지시스템.....	4
[그림 2 -2] TAP 방식의 네트워크 침입탐지시스템.....	4
[그림 2 -3] 인라인 방식의 네트워크 침입탐지시스템.....	5
[그림 2 -4] 단층 퍼셉트론의 구조.....	7
[그림 2 -5] 다층 퍼셉트론의 구조.....	8
[그림 2 -6] 오토인코더의 구조.....	11
[그림 3 -1] 심층신경망 모델의 구조.....	18
[그림 3 -2] 최적 임계치 선정 알고리즘.....	19
[그림 4 -1] 네트워크 트래픽 이상징후 탐지 모델 실험 구조..	21
[그림 4 -2] t-SNE를 통한 데이터 시각화.....	28
[그림 4 -3] 데이터셋의 트래픽 유형별 비율.....	29
[그림 4 -4] C와 gamma의 변화에 따른 F1 스코어.....	30
[그림 4 -5] 심층신경망 모델 통과 후 데이터 시각화.....	32
[그림 4 -6] 기본 오토인코더 모델의 코드 크기별 성능.....	33
[그림 4 -7] 잡음제거 오토인코더 모델의 코드 크기별 성능 ...	34
[그림 4 -8] 학습률에 따른 손실 변화.....	36
[그림 4 -9] 옵티마이저에 따른 손실 변화.....	37
[그림 4 -10] 배치 크기에 따른 성능.....	37
[그림 4 -11] 적층 희소 오토인코더 모델의 복원 오차 분포...	38
[그림 4 -12] 오토인코더 모델별 탐지 성능.....	39
[그림 4 -13] 네트워크 이상징후 탐지 모델의 성능 비교.....	40
[그림 4 -14] 탐지 모델별 새로운 공격 탐지율과 오탐률.....	41

제 1 장 서 론

제 1 절 연구 배경

컴퓨터 및 네트워크 기술이 빠르게 발전함에 따라, 네트워크 트래픽 역시 급격히 증가하고 있다. 2017년에 122.4엑사바이트^①였던 전 세계 월평균 IP 트래픽이 2022년에 396엑사바이트에 이를 것으로 예측된다 [1]. 네트워크 트래픽의 증가와 함께 네트워크를 통한 사이버 위협과 침입 시도 역시 증가하고 있다. 게다가 암호화된 트래픽이 증가하면서 사이버 위협에 대한 식별 및 감시가 더 어려워지고 있다[2].

사이버 위협을 방어하기 위해 대다수 기업은 방화벽(Firewall), 침입탐지시스템(Intrusion Detection System, IDS), 침입방지시스템(Intrusion Prevention System, IPS)^②, 웹 방화벽 등의 하드웨어 또는 소프트웨어 기반의 보안 시스템을 구축하고 있다. 특히, 침입탐지시스템은 비정상적인 네트워크 트래픽과 악의적인 컴퓨터 사용을 실시간으로 탐지하기 위한 시스템으로, 1980년 제임스 앤더슨이 처음 그 개념을 소개[3]한 이후 현재에 이르고 있다. 침입탐지시스템은 탐지 대상에 따라 호스트 침입탐지시스템(Host-based IDS, HIDS)과 네트워크 침입탐지시스템(Network-based IDS, NIDS)으로 구분할 수 있다.

호스트 침입탐지시스템은 컴퓨터의 동작이나 상태를 감시하여 비정상적인 상황이 발생하면 이를 탐지하고 경고를 발생시킨다. 호스트 침입탐지시스템은 호스트마다 상세 분석이 가능하고 사용자 단위의 분석이 가능하다는 장점이 있다. 하지만 네트워크 패킷 단위의 분석이 불가능하고 모든 개별 호스트에서 침입탐지시스템을 운영해야 하므로 자원 소모가 증가할 수 있다.

네트워크 침입탐지시스템은 네트워크상에서 발생하는 트래픽을 분석하여 침입을 탐지한다. 네트워크 침입탐지시스템은 호스트마다 설치할 필요가 없고 전체 네트워크에 대한 분석이 가능하다는 장점이 있다. 하지만 침입탐지시스템을 경유한 공격만 확인이 가능하고 시스템 대상의 공격 시도에 대한 결과를 확인하는 것은 불가능하다.

① 1엑사바이트 = 1,000,000,000GB (10억 기가바이트)

② 침입방지시스템은 능동적 대응 기능이 추가된 침입탐지시스템으로, 침입탐지/방지시스템 모두 핵심 기능은 보안 위협을 찾아내는 것으로 동일함.

침입탐지시스템의 성능을 향상시키기 위해서는 수집된 로그와 네트워크 트래픽을 분석하여 탐지 조건에 맞는 규칙을 수시로 업데이트해야 한다. 하지만 전문가가 아무리 정교한 규칙을 생성한다고 하더라도 공격을 탐지하지 못하거나 잘못 탐지할 가능성은 항상 존재할 수밖에 없다. 이를 보완하기 위해 머신러닝(Machine Learning)을 활용하여 침입탐지를 자동화시키기 위한 연구가 이루어지고 있지만, 실제 보안 시스템에 적용하기에는 아직 부족한 실정이다. 머신러닝 기반의 침입탐지시스템은 데이터 학습을 통해 스스로 침입을 탐지할 수 있지만, 대다수의 경우 학습에 사용할 데이터를 가공하고 정상 트래픽과 공격 트래픽으로 레이블링(Labeling)해야 한다. 게다가 네트워크 트래픽의 특성상 전체 네트워크 트래픽 중 공격 트래픽의 비율은 1%가 채 안 되기 때문에 클래스 불균형에 따른 문제가 발생할 수 있다. 따라서 이러한 문제를 해결할 수 있는 보다 지능적이고 자동화된 네트워크 침입탐지 기법이 필요하다.

제 2 절 연구 내용 및 목적

본 연구에서는 기존 침입탐지시스템의 한계를 극복하고자 자기지도학습 기반의 오토인코더(Autoencoder)를 이용한 네트워크 침입탐지 기법을 제안한다. 오토인코더를 이용하면 정상 트래픽 데이터만 가지고 모델을 학습시키고, 이를 기준으로 공격 트래픽을 탐지해낼 수 있다는 장점이 있다. 따라서 다양한 형태의 오토인코더 모델을 구현하고 성능을 측정하여 최적의 탐지 모델을 선정한다.

본 연구에서 제안한 오토인코더 모델의 적합성을 검증하기 위해, 기존 머신러닝 모델 중 구조적 위험 최소화를 기반으로 일반화 오류를 줄일 수 있는 서포트벡터머신(Support Vector Machine, SVM)을 이용한 탐지 모델의 성능을 측정하고 그 결과를 비교한다. 그리고 딥러닝(Deep Learning) 모델의 기본 구조가 되는 심층신경망(Deep Neural Network, DNN)을 이용한 탐지 모델과도 성능을 비교한다. 또한 기존에 알려지지 않은 새로운 공격 유형에 대한 탐지 성능을 확인하기 위해, 네트워크 트래픽 데이터의 특정 공격 유형을 새로운 공격으로 설정하고 이에 대한 탐지 성능을 확인한다.

제 3 절 연구 보고서 구성

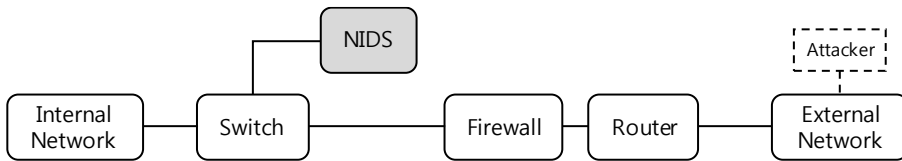
본 연구보고서의 나머지 부분은 다음과 같이 구성된다. 2장에서는 본 연구에 필요한 배경지식과 관련 연구를 설명한다. 3장에서는 본 연구에서 사용한 네트워크 트래픽 데이터셋과 본 연구에서 제안하는 네트워크 이상징후 탐지 모델을 설명한다. 4장에서는 제안한 방식의 탐지 결과를 확인하고 성능을 평가한다. 마지막으로 5장에서 본 연구에 대한 결론을 맺도록 한다.

제 2 장 배경 지식과 관련 연구

제 1 절 네트워크 침입탐지시스템

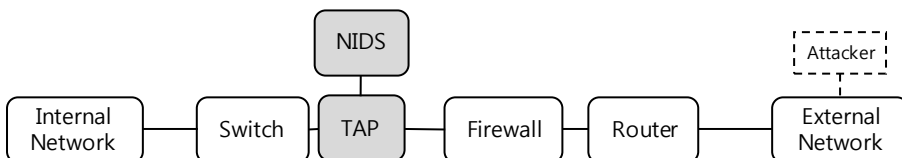
네트워크 침입탐지시스템은 네트워크상에서 발생하는 트래픽을 모니터링하고 사이버 위협을 탐지하는 네트워크 보안 장비이다. 네트워크 침입탐지시스템은 구축하는 방식에 따라서 SPAN (Switched Port Analyzer) 방식, TAP (Terminal Access Point) 방식, 인라인(Inline) 방식으로 분류된다[4].

SPAN 방식의 네트워크 침입탐지시스템은 [그림 2-1]과 같이 구성된다. SPAN은 스위치의 하나 이상의 포트에 유입되는 트래픽의 복사본을 스위치의 다른 모니터링 포트에 전달하는 기술로, 포트 미러링(Port Mirroring)이라고도 한다. 스위치의 SPAN 포트에 네트워크 침입탐지시스템을 연결하고 트래픽 복사본을 대상으로 공격을 탐지한다.



[그림 2-1] SPAN 방식의 네트워크 침입탐지시스템 [4]

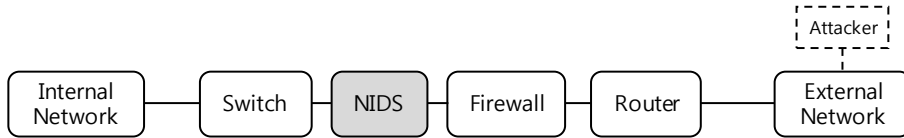
TAP 방식의 네트워크 침입탐지시스템은 [그림 2-2]와 같이 구성된다. TAP은 네트워크 트래픽을 복사하여 전달해주는 네트워크 장비를 말한다. 네트워크망에 TAP 장비 추가하고 이를 네트워크 침입탐지시스템에 연결하여 공격을 탐지한다.



[그림 2-2] TAP 방식의 네트워크 침입탐지시스템 [4]

인라인 방식의 네트워크 침입탐지시스템은 [그림 2-3]과 같이 구성

된다. 인라인 방식은 네트워크 방화벽 구성과 동일하게 모든 트래픽이 네트워크 침입탐지시스템을 거쳐서 갈 수 있도록 하는 방식이다. 인라인 방식은 공격 트래픽이 탐지되면 바로 차단할 수 있지만, 네트워크 성능이 저하되고 하드웨어 장애가 발생하면 네트워크 장애까지 발생하는 단점이 있다.



[그림 2-3] 인라인 방식의 네트워크 침입탐지시스템 [4]

대부분의 네트워크 침입탐지시스템은 SPAN 방식이나 인라인 방식으로 구성한다. 이러한 네트워크 침입탐지시스템은 탐지 방식에 따라 시그니처(Signature-based) 탐지와 이상징후(Anomaly-based) 탐지로 구분할 수 있다.

1. 시그니처 탐지

시그니처 탐지 방식은 사전에 정의한 공격 패턴을 기반으로 공격 트래픽을 탐지하는 방식이다. 기존 네트워크 침입에 대한 트래픽 분석 결과를 바탕으로 공격 패턴을 설정하고, 발생한 트래픽이 이 패턴과 일치하는 경우 공격으로 판단한다. 이로 인해 시그니처 탐지 방식은 이미 알려진 공격에 대해서는 신속하고 효율적으로 탐지해내지만, 패턴에 없는 새로운 유형의 공격이 발생하거나 시그니처 정보가 일부 변경되면 즉각적인 탐지 및 대처가 불가능하다[5].

스노트(Snort) [6]는 대표적인 시그니처 탐지 기반의 네트워크 침입 탐지시스템으로, 패턴 매칭 알고리즘을 사용한 네트워크 침입탐지 기능을 제공한다. 스노트는 스니퍼(Sniffer), 전처리기(Preprocessor), 탐지 엔진(Detection Engine), 경고/로깅(Alerts/Logging)의 네 가지 구성요소로 이루어진다. 스니퍼에서 수집한 네트워크 패킷은 전처리 작업을 거친 뒤 탐지 엔진에서 해당 패킷의 헤더 정보와 페이로드, 패킷 사이즈 등을 등록된 시그니처와 매칭하여 공격 여부를 판단한다. 스노트의 핵심 모듈인 탐지 엔진은 사전에 정의한 공격 패턴을 기반으로 동작하기 때문에 정확한 룰 분석 및 운영이 요구되고, 공격 탐지 룰이 많아질수록 성

능이 저하된다는 단점이 있다.

2. 이상징후 탐지

이상징후 탐지 방식은 네트워크 트래픽의 정상 동작을 기반으로 비정상적인 트래픽을 탐지해내는 방식으로, 새로운 공격 트래픽에 대한 탐지가 가능하다. 하지만 이상징후 탐지 방식은 정상 트래픽을 공격 트래픽으로 오탐하는 비율이 높는데, 그 이유는 이전에 확인되지 않았던 새로운 네트워크 트래픽의 동작을 비정상 동작으로 인식할 수 있기 때문이다.

이상징후 탐지를 위한 기본적인 방법론으로는 주성분 분석, 혼합 모델 등의 통계적 기법, 유사도, 거리, 밀도, 그래프 기반의 군집화 기법, 엔트로피를 이용한 정보이론 기반 탐지 기법, 머신러닝 기법 등이 있다 [7]. 특히, 대량의 네트워크 트래픽 데이터를 바탕으로 모델을 학습하여 공격 패턴을 추출하고 탐지할 수 있는 다양한 머신러닝 기법이 연구되어 왔다. Kim 등[8]은 서포트벡터머신 기반 탐지 모델에 유전 알고리즘을 적용하여 공격 트래픽에 대한 탐지 성능을 높였다. Jalil 등[9]은 신경망(Neural Network), 서포트벡터머신, 의사결정트리(Decision Tree)를 이용한 탐지 모델의 성능을 비교하고, 의사결정트리 모델이 다른 두 모델보다 높은 성능을 보임을 확인했다. Meng[10]은 실험을 통해 이상징후 탐지 모델의 성능은 실험 환경과 설정에 강한 의존도를 나타낸다는 것을 확인했다. 따라서 머신러닝 기법을 실제 운영 환경에 적절한 방식으로 적용하여 탐지 성능을 향상시킬 필요가 있다.

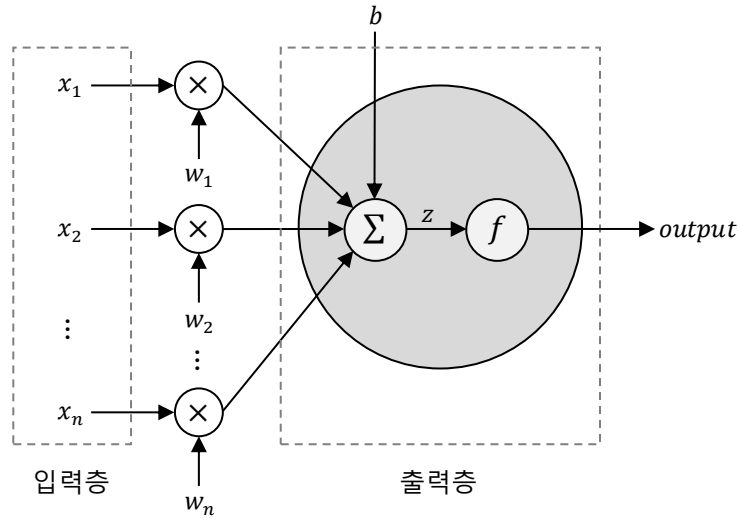
최근 딥러닝이 다양한 분야에서 우수한 성능을 나타내면서, 네트워크 이상징후 탐지 모델에도 딥러닝을 활용하는 연구가 활발히 이루어지고 있다. 이에 대한 내용은 본 장 3절에서 다루도록 한다.

제 2 절 딥러닝 모델

1. 심층신경망

심층신경망은 딥러닝의 가장 기본적인 학습 모델로서, 입력층(Input Layer)과 출력층(Output Layer) 사이에 2개 이상의 은닉층(Hidden Layer)을 포함하고 있는 인공신경망(Artificial Neural Network, ANN)을 말한다.

심층신경망을 구성하는 기본 단위는 퍼셉트론(Perceptron)이다. 퍼셉트론은 1957년 Frank Rosenblatt에 의해 고안된 선형 분류 모델[11]로, [그림 2-4]는 단층 퍼셉트론의 구조를 나타낸다. x 는 입력 벡터의 값을 나타내고, w 는 가중치(Weight), b 는 편향(Bias), f 는 활성화 함수(Activation Function)를 나타낸다.



[그림 2-4] 단층 퍼셉트론의 구조

단층 퍼셉트론은 입력층과 출력층으로 구성되며, 입력층을 통해 입력된 데이터는 출력층 뉴런으로 전달되고 활성화 함수를 거쳐 값이 출력된다. 계단 함수를 활성화 함수로 사용한 퍼셉트론을 수식으로 나타내면 식(1)과 같다.

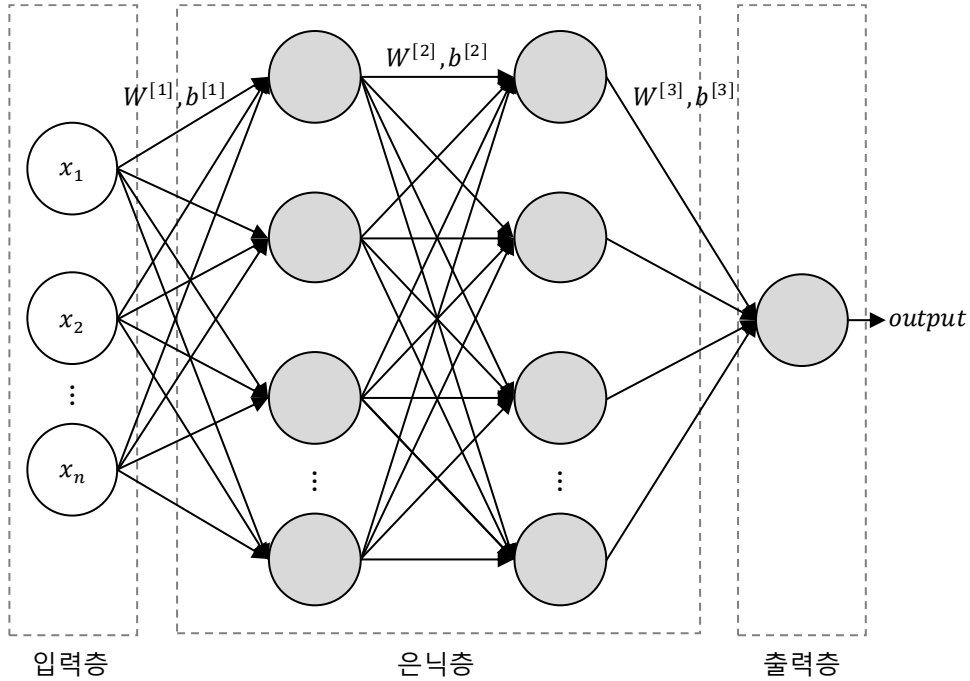
$$output = f\left(\left(\sum_{i=1}^n w_i x_i\right) + b\right) \text{ where } f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \quad (1)$$

단층 퍼셉트론은 선형 분류만 가능하기 때문에 기본적인 XOR (Exclusive-OR) 게이트^③ 로직조차 구현할 수 없다는 한계점이 있다.

다층 퍼셉트론(Multilayer Perceptron, MLP)은 단층 퍼셉트론의 XOR 문제를 해결하기 위해 은닉층에 비선형성을 도입한 모델이다. [그

^③ XOR 게이트는 입력값이 서로 다르면 1을 출력하고, 같으면 0을 출력함.

림 2-5]는 2개의 은닉층을 가진 다층 퍼셉트론의 구조를 나타낸다.



[그림 2-5] 다층 퍼셉트론의 구조

초기의 다층 퍼셉트론은 은닉층 뉴런의 활성화 함수로 시그모이드 (Sigmoid) 함수를 사용했다. 시그모이드는 S자 모양이라는 의미로, 아래 식(2)와 같이 미분이 용이한 형태를 주로 사용한다.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2)$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

각 층의 가중치와 편향 값은 역전파(Backpropagation) 알고리즘[12]을 통해 학습한다. 먼저 가중치와 편향 값을 임의의 값으로 초기화하고, 입력 데이터에 대한 출력값을 확인한다. 출력값과 목표값 사이의 오차를 구하고, 경사 하강법(Gradient Descent)을 이용해 오차가 감소하는 방향으로 출력층에서 입력층까지 가중치와 편향 값을 업데이트한다. 하지만 시그모이드 함수를 사용하면 은닉층의 개수가 늘어날수록 이전 계층으로 전달되는 값이 현저하게 작아지게 되고, 결국 출력층의 오차가 입

력층까지 전달되지 않는 기울기 소실(Vanishing Gradient)이 발생할 수 있다. 시그모이드 함수는 0과 1 사이의 값만 반환하므로, 체인 룰(Chain Rule)을 이용해 계속 값을 곱해나가면 그 값이 0으로 수렴할 수 밖에 없기 때문이다. 이 문제를 해결하기 위해 ReLU (Rectified Linear Unit) 함수[13]가 제안되었고, 이후 많은 종류의 활성화 함수가 연구되었다. 은닉층에서 주로 사용되는 활성화 함수의 종류는 [표 2-1]과 같다[14].

[표 2-1] 활성화 함수 종류 [14]

활성화 함수	수식
Sigmoid	$\sigma(z) = \frac{1}{1 + e^{-z}}$
Tanh	$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
ReLU	$\max\{0, z\}$
Leaky ReLU	$\max\{0.01z, z\}$
ELU	$\begin{cases} z & z \geq 0 \\ \alpha(e^z - 1) & z < 0 \end{cases}$
Maxout	$\max\{z_1, z_2\}$

출력층에서 사용하는 함수는 이진 분류(Binary Classification)와 다중 클래스 분류(Multiclass Classification)에 따라 구분된다. 이진 클래스 분류에서는 시그모이드 함수를 사용하고, 다중 클래스 분류에서는 소프트맥스(Softmax) 함수를 사용한다. 소프트맥스 함수는 시그모이드 함수를 일반화한 것으로 수식은 식 (3)과 같다. 소프트맥스 함수의 출력은 0과 1 사이의 값이며, 출력값들의 총합은 1이 된다. 이러한 성질로 인해 함수의 출력을 확률로 해석할 수 있고, 다중 분류가 가능하다.

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (3)$$

심층신경망 학습 시, 학습 상태를 측정하는 지표로 손실함수(Loss

Function)를 사용한다. 대표적인 손실함수는 평균제곱오차(Mean Squared Error, MSE)와 교차엔트로피오차(Cross Entropy Error, CEE)가 있다. 평균제곱오차는 식(4)와 같으며, y 는 신경망의 출력값, t 는 목표값, k 는 데이터의 차원 수를 의미한다.

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2 \quad (4)$$

교차엔트로피오차는 식(5)와 같으며, \log 는 밑이 e 인 자연로그(\log_e), y 는 신경망의 출력값, t 는 목표값, k 는 데이터의 차원 수를 의미한다.

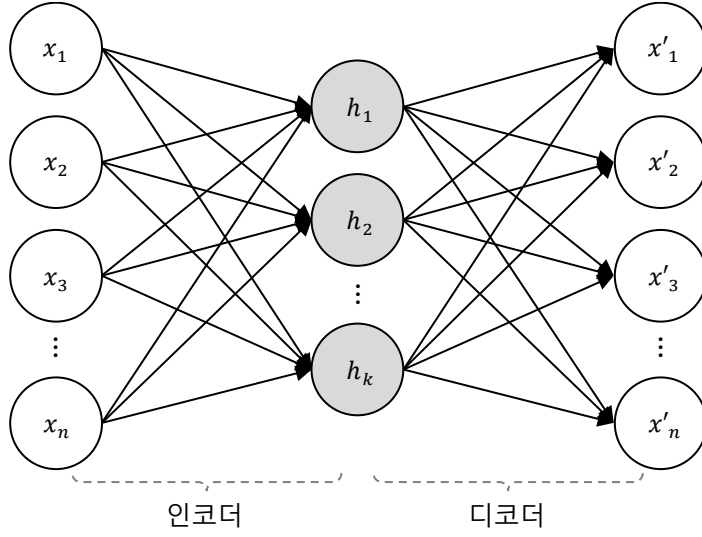
$$E = - \sum_k t_k \log y_k \quad (5)$$

심층신경망은 일반적으로 2개 이상의 은닉층을 포함한 다층 퍼셉트론을 의미한다. 고차원의 데이터 분류가 가능한 심층신경망은 이미지 처리 분야에서 높은 성능을 보이는 합성곱신경망(Convolutional Neural Network, CNN)과 자연어 처리 분야에서 높은 성능을 보이는 순환신경망(Recurrent Neural Network, RNN) 등으로 발전하여 다양한 분야에서 활용되고 있다.

본 연구에서는 심층신경망 기반의 네트워크 이상징후 탐지 모델을 구현하고 네트워크 공격 트래픽에 대한 탐지 성능을 확인한다.

2. 오토인코더

오토인코더는 자기지도학습에 속하는 인공신경망의 한 형태로, 신경망을 거쳐 나온 출력값이 원래의 입력값과 동일해지도록 학습하는 모델이다[15]. 즉, 입력 데이터와 레이블을 가지고 학습하는 것이 아니라 입력 데이터만을 이용하여 데이터 자체의 특징을 압축하고 다시 복원한다. 오토인코더는 인코더(Encoder)와 코드(Code), 디코더(Decoder)로 구성되어 있으며, 기본 구조는 [그림 2-6]과 같다.



[그림 2-6] 오토인코더의 구조

오토인코더는 일반적으로 코드층(Code Layer)의 크기를 입력층의 크기보다 작게 설정하여 입력 데이터의 주요 특성을 학습할 수 있도록 한다. 인코더는 식(6)과 같이 고차원의 입력 데이터를 저차원의 잠재 공간에 매핑(Mapping)하고, 디코더는 식(7)과 같이 압축된 저차원의 데이터를 다시 고차원의 데이터로 복원(Reconstruction)한다.

$$h = f_{\theta}(x) = a(Wx + b) \quad (6)$$

$$x' = g_{\theta'}(h) = a(W'h + b') \quad (7)$$

위 식에서 θ 는 인코더의 파라미터 $\theta = \{W, b\}$, θ' 는 디코더의 파라미터 $\theta' = \{W', b'\}$, $a(\cdot)$ 는 활성화 함수를 의미한다[16]. 오토인코더는 기본적으로 복원 오차(Reconstruction Error)를 최소화하도록 모델을 최적화하며, 이는 식(8)과 같이 나타낼 수 있다. L 은 손실함수를 의미하며, 평균제곱오차와 교차엔트로피오차를 주로 사용한다.

$$\begin{aligned} \theta^*, \theta'^* &= \underset{\theta, \theta'}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n L(x^{(i)}, x'^{(i)}) \\ &= \underset{\theta, \theta'}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n L(x^{(i)}, g_{\theta'}(f_{\theta}(x^{(i)}))) \end{aligned} \quad (8)$$

오토인코더는 위와 같은 기본적인 형태 외에도 잡음제거 오토인코더(Denoising Autoencoder), 희소 오토인코더(Sparse Autoencoder), 적층 오토인코더(Stacked Autoencoder)가 있다.

잡음제거 오토인코더는 데이터에 잡음(Noise)을 추가하여 모델의 입력으로 사용하고, 잡음이 없는 원래 데이터로 복원하도록 모델을 학습시키는 구조이다. 이를 통해 오토인코더가 데이터의 주요 특성을 학습하도록 한다. 일반적으로 입력 데이터에 직접 가우시안 잡음을 추가하거나 드롭아웃(Dropout) 계층을 추가하여 사용한다.

희소 오토인코더는 데이터의 주요 특성을 학습하기 위해 희소성(Sparsity)을 이용한다[17]. 입력층의 크기보다 코드층의 크기가 작은 기본적인 오토인코더의 형태와 달리, 입력층의 크기보다 코드층의 크기를 크게 설정한다. 대신 희소 제약을 추가하여 뉴런이 크게 활성화되지 않도록 규제하고 데이터의 주요 특성을 학습하도록 한다.

적층 오토인코더는 여러 층의 은닉층을 가지는 오토인코더이며, 은닉층을 추가할수록 더 복잡한 특징을 학습할 수 있다. 적층 오토인코더는 일반적으로 코드층을 중심으로 인코더와 디코더가 대칭인 구조를 가진다.

본 연구에서는 다양한 형태의 오토인코더 기반의 네트워크 이상징후 탐지 모델을 구현하고, 네트워크 공격 트래픽에 대한 탐지 성능을 확인한다.

제 3 절 딥러닝 기반 네트워크 이상징후 탐지 연구

딥러닝은 다수의 비선형 은닉층을 포함하고 있는 신경망으로, 복잡한 특징 공간을 모델링할 수 있는 기술이다. 딥러닝은 컴퓨터 비전, 자연어 처리, 음성 인식, 시계열 분석과 같은 다양한 영역에서 우수한 성능을 내고 있으며[18], 최근 네트워크 침입탐지에 딥러닝을 활용하는 연구가 활발히 이루어지고 있다.

Staudemeyer 등[19]의 연구에서는 장단기메모리(Long Short-Term Memory, LSTM) 순환신경망을 이용한 네트워크 침입탐지시스템을 제안했다. KDD Cup 1999 데이터셋[20]을 사용하여 장단기메모리 순환신경망이 학습 데이터에 숨겨진 모든 공격 유형을 학습할 수 있음을 보였다. Gao 등[21]은 심층신경신뢰망(Deep Belief Network, DBN)을 이용한 침입탐지시스템을 연구했다. KDD Cup 1999 데이터셋을 사용해

모델의 성능을 측정했고, 서포트벡터머신과 인공신경망 모델보다 높은 탐지 성능을 보임을 확인했다. Kim 등[22]의 연구에서는 침입탐지시스템에 심층신경망 알고리즘을 적용하고, KDD Cup 1999 데이터셋을 사용하여 검증했다. 제안한 모델은 정확도, 재현율, 오탐률 측면에서 기존 모델보다 우수한 성능을 나타냈다. Le 등[23]은 침입탐지시스템에서 널리 사용되는 서포트벡터머신, k-최근접이웃(k-Nearest Neighbor, kNN)이 높은 오탐률을 보임을 밝히고, 이를 개선하기 위해 장단기메모리 순환신경망 모델을 침입탐지시스템에 적용했다. 여섯 종류의 옵티마이저(Optimizer) 중 모델에 가장 적합한 것을 찾고, 그 중 Nadam을 사용했을 때 이전 연구보다 탐지 성능이 향상됨을 보였다. 하지만 위 연구들에서 사용한 KDD Cup 1999 데이터셋은 중복 레코드들로 인해 상당히 편향된 결과를 나타낼 수 있다고 알려져 있다[24].

Javaid 등[25]은 KDD Cup 1999 데이터셋을 개선한 NSL-KDD 데이터셋[26]을 사용하여 딥러닝 기반의 Self-taught Learning (STL) 기법을 제안했다. 비지도 특징 학습(Unsupervised Feature Learning, UFL)^④ 중 희소 오토인코더(Sparse Autoencoder)를 이용하여 모델을 학습시키고, 사전 학습(Pre-training)된 인코더에 소프트맥스 회귀(Softmax Regression)을 결합하여 세부 조정(Fine Tuning) 후 공격을 탐지했다. Yousefi-Azar 등[27]의 연구에서는 제한된 볼츠만머신(Restricted Boltzmann Machine, RBM)을 사전 학습 후, 가우시안 나이브베이지스(Gaussian Naive Bayes), k-최근접이웃, 서포트벡터머신, XGBoost 모델을 사용하여 공격을 탐지했다. Li 등[28]은 합성곱신경망을 이용한 네트워크 침입탐지시스템을 제안했다. 트래픽 데이터를 8×8 이미지 형태의 2차원 벡터로 변환한 뒤, ResNet-50과 GoogLeNet을 이용하여 학습시켰다. 그리고 기존 머신러닝 모델과 성능 비교를 통해 제안한 딥러닝 모델이 우수한 성능을 나타냄을 확인했다. 위 연구들에서 사용한 NSL-KDD 데이터셋 역시 KDD Cup 1999 데이터셋을 기반으로 하고 있기 때문에 최신 네트워크 침입 패턴을 반영하기 어렵고, 시물레이션 네트워크 환경에서 수집된 데이터이기 때문에 실제 네트워크 트래픽을 완벽하게 표현할 수 없다.

Vinayakumar 등[29]은 KDD Cup 1999, NSL-KDD 데이터셋 뿐만 아니라, UNSW-NB15, Kyoto, CICIDS2017 등 비교적 최근에 수집된

^④ 비지도 특징 학습의 주요 기법에는 희소 오토인코더, 제한된 볼츠만머신, k-평균 군집화, 가우시안 혼합 모델 등이 있음.

네트워크 트래픽 데이터셋을 연구에 사용했다. 심층신경망 기반의 네트워크 침입탐지 모델을 통해 트래픽 데이터의 추상적이고 고차원적인 특징 표현을 학습하고, 그 결과 기존 머신러닝 모델보다 성능이 우수함을 확인했다. 하지만 위 연구에서는 학습데이터와 평가데이터를 분리할 때 새로운 공격 유형에 대해 고려하지 않아서 새로운 공격 유형에 대한 탐지 성능은 확인할 수 없었다.

본 연구에서는 기존 네트워크 트래픽 데이터셋의 문제점을 개선한 CICIDS2017 데이터셋[30]을 사용하여 연구를 진행하고자 한다. 기존 연구에서 주로 사용해온 지도학습 기반의 모델 대신 자기지도학습 기반의 오토인코더를 이용하여 네트워크 침입탐지 모델을 구현하고, 모델의 탐지 성능을 확인한다. 또한 기존에 알려지지 않은 새로운 공격에 대한 탐지 상황을 설정하고 그에 대한 성능을 확인하고자 한다.

제 3 장 연구 방법

제 1 절 네트워크 트래픽 데이터셋

네트워크 트래픽 이상징후 탐지 연구에서는 신뢰성 있는 데이터셋을 확보하는 것이 중요하다. 네트워크 트래픽의 다양성과 샘플 수, 수집 환경 등이 고려된 네트워크 트래픽 데이터셋을 사용해야 한다.

KDD Cup 1999 데이터셋[20]은 1998년 DARPA (Defense Advanced Research Projects Agency) 침입탐지 평가 프로그램에서 수집한 데이터^⑤를 기반으로 만들어진 데이터셋으로, 그동안 네트워크 보안 분야에서 널리 사용되어 왔다. 이 데이터셋에는 38가지의 공격^⑥이 포함되어있고, 이를 DoS (Denial of Service), R2L (Remote To Local), U2R (User To Root), Probing과 같이 4가지 공격 유형으로 분류하고 있다. 하지만 KDD Cup 1999 데이터셋은 수많은 중복 레코드와 일부 공격 패턴에 대한 정보 부족 등의 문제로 인해 평가 결과에 대한 신뢰성을 보장할 수 없다. Tavallaee 등[26]은 이러한 문제점을 개선한 NSL-KDD 데이터셋을 제안했다. 하지만 NSL-KDD 데이터셋 역시 KDD Cup 1999 데이터셋을 기반으로 하고 있기 때문에 최신 네트워크 침입 패턴을 반영하지 못하고, 실제 네트워크 환경에서 수집된 데이터가 아니기 때문에 실제 네트워크 트래픽을 완벽하게 표현할 수 없다는 한계점이 있다.

반면, 캐나다 뉴 브런스윅 대학교의 CIC (Canadian Institute for Cybersecurity)에서 제공한 CICIDS2017 데이터셋[30]은 2017년 7월에 수집한 데이터로, 최신 네트워크 침입 시나리오까지도 포함하고 있다. I. Sharafaldin 등[31]은 HTTP, HTTPS, FTP, SSH 및 Email 프로토콜 기반의 사용자 동작을 구축한 B-프로파일 시스템을 이용하여 실제 네트워크 환경을 구축하고 데이터를 생성 및 수집했다. CICIDS2017 데이터셋은 Brute Force, Heartbleed, Botnet, DoS(Denial-of-Service), DDoS(Distributed DoS), Web Attack, Infiltration 등 다양한 공격 시나

^⑤ 1998 DARPA 데이터셋은 MIT 링컨 연구실에 의해 U.S. Air Force LAN의 모의 환경에서 9주 동안 수집됨.

^⑥ 학습 데이터는 24가지의 공격으로 구성되어있고, 평가 데이터는 14가지의 공격이 추가로 포함되어 있음.

리오를 바탕으로 총 5일^⑦에 걸쳐 수집되었다.

[표 3-1]은 네트워크 트래픽 데이터셋의 신뢰성을 평가하기 위해 10가지 기준으로 KDD Cup 1999, NSL-KDD, CICIDS2017을 비교한 결과이다[30]. 본 연구에서는 아래 기준을 모두 만족하는 CICIDS2017 데이터셋을 사용하여 실험을 진행하도록 한다.

[표 3-1] 네트워크 트래픽 데이터셋 비교 [30]

기준		KDD Cup 1999	NSL-KDD	CICIDS2017
전체 네트워크 구성		예	예	예
실제 트래픽		아니오	아니오	예
레이블 데이터		예	예	예
네트워크 상호작용		예	예	예
전체 트래픽 캡처		예	예	예
사용 가능한 프로토콜	HTTP	예	예	예
	HTTPS	아니오	아니오	예
	SSH	예	예	예
	FTP	예	예	예
	Email	예	예	예
공격 다양성	Browser	아니오	아니오	예
	Bruteforce	예	예	예
	DoS	예	예	예
	Scan	예	예	예
	Backdoor	아니오	아니오	예
	DNS	아니오	아니오	예
	기타 공격	예	예	예
네트워크 도메인 이질성		아니오	아니오	예
데이터 특성 정보 제공		예	예	예
메타정보 제공		예	예	예

^⑦ 2017년 7월 3일부터 2017년 7월 7일까지 수집함.

제 2 절 네트워크 이상징후 탐지 모델

1. 서포트벡터머신 기반 탐지

서포트벡터머신은 머신러닝의 대표적인 분류 모델로서, 두 범주를 구성하는 데이터들을 분리할 수 있는 최적의 초평면(Hyperplane)을 찾아내는 이진 선형 분류 모델이다[32]. 서포트벡터머신은 경험적 위험 최소화 원칙(Empirical Risk Minimization)을 기반으로 하는 일반적인 머신러닝 모델과는 달리 구조적 위험 최소화(Structural Risk Minimization)를 기반으로 하여 우수한 일반화 성능을 보여준다[33]. 또한 데이터를 고차원 특징 공간으로 사상하는 커널 트릭(Kernel Trick)을 사용하여 비선형 분류를 할 수 있는 모델이다.

본 연구에서는 가우시안 방사 기저 함수(Radial Basis Function, RBF)를 커널 함수로 사용한 서포트벡터머신을 이용하여 네트워크 이상징후를 탐지한다. [표 3-2]의 파라미터를 기준으로 그리드 서치(Grid Search)방식을 사용하여 모델에 적합한 C와 gamma 값을 찾는다.

[표 3-2] 서포트벡터머신 모델의 파라미터

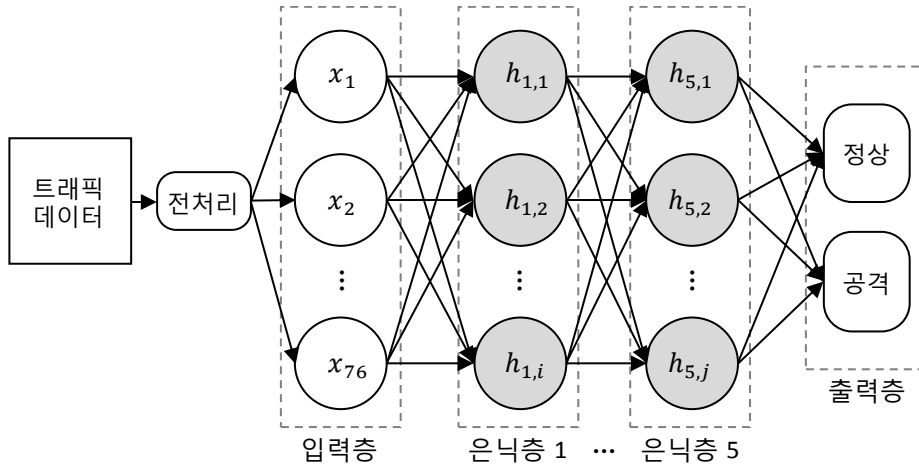
파라미터	값
C	0.01, 0.1, 1, 10, 100
gamma	0.01, 0.1, 1, 10, 100

C는 얼마나 많은 데이터가 다른 클래스에 놓이는 것을 허용하는지를 결정한다. C가 너무 낮으면 과소적합, C가 너무 높으면 과대적합이 발생할 수 있다. gamma는 데이터가 영향력을 행사하는 거리를 결정한다.

2. 심층신경망 기반 탐지

본 연구에서는 네트워크 트래픽 데이터의 복잡한 비선형 관계를 모델링할 수 있도록 지도학습 기반의 심층신경망을 이용하여 네트워크 이상징후를 탐지한다. [그림 3-1]은 본 연구에서 이용할 심층신경망 모델의 구조를 보여준다. 입력층은 전처리된 입력 데이터의 차원과 일치하도록 76개의 뉴런으로 구성하고, 각 뉴런들은 입력층에서 은닉층으로, 은닉층에서 출력층으로 완전히 연결된 구조를 사용한다. 기울기 소실을 방지하

기 위해 은닉층의 활성화 함수는 ReLU를 사용하고, 출력층에는 시그모이드 함수를 사용하여 데이터가 정상 트래픽인지 공격 트래픽인지 여부를 판단한다. 그리고 모델이 학습 데이터에 과적합되는 것을 방지하기 위하여 각 은닉층에 드롭아웃을 적용한다. 손실 함수는 교차엔트로피오차를 사용하고, Adam 옵티마이저를 사용하여 학습을 진행한다.



[그림 3-1] 심층신경망 모델의 구조

심층신경망의 하이퍼 파라미터는 [표 3-3]의 값을 기준으로 시행착오를 통해 최적의 모델을 구성한다.

[표 3-3] 심층신경망 모델의 하이퍼 파라미터 설정

하이퍼 파라미터	종류/값
은닉층 수	2, 3, 4, 5
은닉층별 뉴런 수	4, 8, 16, 32, 64, 128
활성화 함수	ReLU
출력 함수	Sigmoid
드롭아웃	0.2
손실 함수	Cross Entropy
옵티마이저	Adam
배치 사이즈	256
학습 횟수	500
학습률	0.001

3. 오토인코더 기반 탐지

본 연구에서는 자가지도학습 기반의 오토인코더를 이용한 네트워크 이상징후 탐지 모델을 제안한다. 기존 지도학습 기반의 네트워크 이상징후 탐지의 경우, 정상 트래픽과 공격 트래픽으로 레이블링된 데이터셋이 필요하다. 하지만 실제 네트워크 트래픽 수집 시 데이터에 일일이 레이블링 작업을 하기는 쉽지 않다. 또한 네트워크 트래픽 특성상 전체 네트워크 트래픽 중 공격 트래픽의 비율이 1%가 채 안 되기 때문에 클래스 불균형으로 인해 탐지 모델을 제대로 학습시키기 어려울 수 있다. 따라서 본 연구에서는 정상 트래픽 데이터만 사용하여 모델을 학습시킬 수 있는 오토인코더를 이용하여 네트워크 이상징후를 탐지한다.

정상 트래픽 데이터로 학습시킨 오토인코더 모델은 정상 트래픽에 대해서는 낮은 복원 오차를 나타내지만, 공격 트래픽에 대해서는 높은 복원 오차를 나타낸다. 따라서 정상 트래픽만 가지고 모델을 학습시키고 복원 오차에 대한 최적의 임계치(Threshold) 값을 찾은 뒤, 이를 기준으로 공격 데이터를 탐지하도록 한다.

Algorithm find optimal threshold

Input: reconstruction error of benign traffic data: E_{benign}

Output: optimal threshold

```

1:  $f1score_{max} \leftarrow 0$ 
2:  $threshold_{max} \leftarrow 0$ 
3: calculate  $\mu_{re}$  and  $\sigma_{re}$  of  $E_{benign}$ 
4:  $threshold \leftarrow \mu_{re} + 3\sigma_{re}$ 
5: while  $threshold > \mu_{re}$  do
6:   calculate  $f1score$  of the model using  $threshold$ 
7:   if  $f1score > f1score_{max}$  then
8:      $f1score_{max} \leftarrow f1score$ 
9:      $threshold_{max} \leftarrow threshold$ 
10:   $threshold \leftarrow threshold - 0.01$ 
11: end while
12: return  $threshold_{max}$ 

```

[그림 3-2] 최적 임계치 선정 알고리즘

[그림 3-2]는 오토인코더 모델의 복원 오차를 가지고 최적의 임계치를 찾는 과정을 보여준다. 정상 트래픽에 대한 복원 오차의 평균과 표준편차를 구하고, 그 값을 이용하여 임계치를 설정한다. 해당 임계치를 기

준으로 탐지한 결과에 대해 F1 스코어(F1 score)를 확인한다. 이후 임계치 값을 0.01씩 감소시키면서 F1 스코어가 최대가 되는 임계치 값을 찾는다.

오토인코더 모델의 입력층과 출력층의 뉴런 수는 데이터의 차원과 동일하게 구성한다. 은닉층은 이보다 적은 수의 뉴런으로 구성하고, 실험을 통해 최적의 구성을 찾는다. 각 층의 활성화 함수는 기본적으로 ReLU를 사용한다. 단, 입력 데이터가 최소-최대 정규화(Min-Max Normalization)를 통해 0과 1 사이의 값을 가지므로 출력층의 함수는 시그모이드를 사용한다. [표 3-4]의 하이퍼 파라미터를 기준으로 시행착오를 통해 최적의 모델을 제안한다.

[표 3-4] 오토인코더 모델의 하이퍼 파라미터 설정

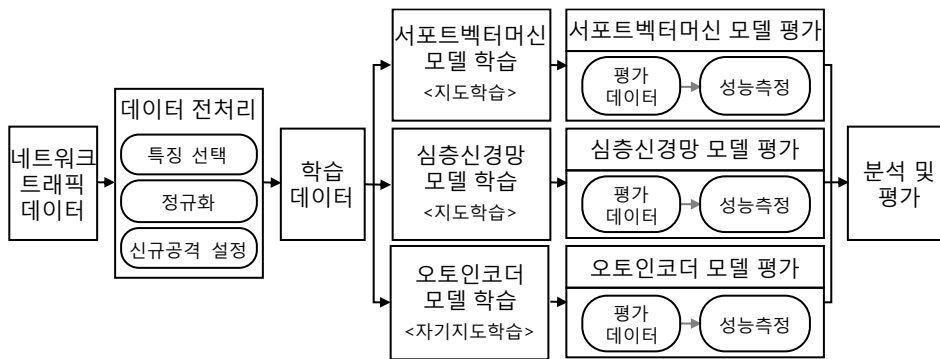
하이퍼 파라미터	종류/값
코드 크기	4, 8, 16, 32, 64
인코더 은닉층 수	1, 2, 3
활성화 함수	ReLU
출력 함수	Sigmoid
손실 함수	Mean Squared Error
옵티마이저	Adam
배치 사이즈	256
학습 횟수	500
학습률	0.001

제 4 장 실험 및 결과

제 1 절 실험 개요 및 환경

1. 실험 개요

본 실험에서는 지도학습 기반의 서포트벡터머신과 심층신경망, 자기 지도학습 기반의 오토인코더를 이용한 네트워크 이상징후 탐지 모델을 구현하고, 각 모델의 탐지 성능을 측정한다. 그리고 성능 평가 지표를 바탕으로 이상징후 탐지 모델 간의 성능을 비교하고 그 효과를 확인한다.



[그림 4 - 1] 네트워크 트래픽 이상징후 탐지 모델 실험 구조

2. 실험 환경

본 연구의 실험은 Python 3.6.8 프로그래밍 언어를 사용하여 진행한다. 네트워크 트래픽 데이터의 전처리를 위해 Numpy 1.16.1, Pandas 0.24.1 라이브러리를 사용하고, 데이터 정규화 및 학습 데이터와 평가 데이터 분리는 Scikit-learn 0.20.2 라이브러리를 사용한다. 딥러닝 모델의 구현은 Tensorflow 1.12.0과 Tensorflow를 백엔드(Back-end)로 하는 Keras 2.2.4를 사용한다. 모델의 학습 및 성능 평가는 Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz, GeForce RTX 2080 Ti 환경에서 진행한다. [표 4 - 1]은 본 연구의 실험 환경 정보를 나타낸다.

[표 4-1] 실험 환경 정보

실험 환경	설명
CPU	Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz x 8
RAM	64 GB
GPU	GeForce RTX 2080 Ti
OS	Ubuntu 16.04 LTS
GPGPU API	CUDA Toolkit 9.0 / cuDNN 7.4.2

3. 성능 평가 지표

본 연구에서 제안하는 모델의 성능을 평가하기 위해서는 네트워크 트래픽 데이터의 특성에 적합한 성능 평가 지표를 선정하는 것이 중요하다. 본 연구에서는 모델의 성능을 평가하기 위해 오차행렬(Confusion Matrix)을 사용한다. 오차행렬을 통해서 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1 스코어(F1 score)를 계산하고 이를 바탕으로 모델별 성능을 비교 분석한다[34].

오차행렬은 데이터셋에 대해 모델이 분류한 결과를 나타내는 표로, 네트워크 이상징후 탐지 모델의 오차행렬은 [표 4-2]와 같이 나타낼 수 있다.

[표 4-2] 네트워크 트래픽 데이터의 오차행렬

실제 트래픽 클래스	모델이 예측한 트래픽 클래스	
	정상 트래픽	공격 트래픽
정상 트래픽	TN (True Negative)	FP (False Positive)
공격 트래픽	FN (False Negative)	TP (True Positive)

정확도는 예측 결과와 실제 값이 얼마나 동일한지에 대한 비율로, 전체 네트워크 트래픽 중 정상 트래픽과 공격 트래픽을 정확히 예측한 비율을 나타낸다. 정확도는 오차행렬 상에서 수식으로 나타내면 식(9)와 같다.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

정확도는 분류 모델의 성능을 직관적으로 나타내는 지표이지만, 불균형한 데이터셋에서는 모델의 성능을 왜곡시킬 수 있다. 네트워크 트래픽 데이터의 경우, 정상 트래픽의 양이 공격 트래픽의 양보다 압도적으로 많기 때문에 모든 데이터를 정상 트래픽으로 예측하더라도 정확도는 높게 나올 수 있다. 그래서 불균형한 데이터셋에서는 정확도보다 정밀도와 재현율을 더욱 신뢰할 수 있다.

정밀도는 모델이 양성으로 예측한 데이터 중 실제 값이 양성인 데이터의 비율을 뜻하며, 공격 트래픽으로 예측한 트래픽 중 실제 공격 트래픽의 비율을 나타낸다. 이는 식(10)과 같다.

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

재현율은 실제 값이 양성인 데이터 중 모델이 양성으로 예측한 데이터의 비율을 뜻한다. 실제 공격 트래픽 중 공격 트래픽으로 예측한 비율을 나타내며, 식(11)과 같다. 본 연구에서는 재현율을 통해 새로운 공격 유형에 대한 탐지 성능을 검증한다.

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

정밀도와 재현율은 상호 보완적인 평가 지표이므로, 어느 한쪽의 수치를 강제로 높이면 다른 한쪽의 수치는 떨어질 가능성이 있다. F1 스코어는 식(12)와 같이 정밀도와 재현율의 조화 평균을 의미하며, 불균형 클래스에서 정확한 평가를 위해 주로 사용되는 지표이다. F1 스코어가 높을수록 모델의 성능이 좋다는 것을 의미한다. 본 연구에서 제안하는 모델의 탐지 성능은 F1 스코어를 이용하여 검증한다.

$$F1 \text{ score} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (12)$$

또한 네트워크 보안 분야에서 정상 트래픽을 공격 트래픽으로 파악하는 오탐이 큰 문제가 되므로, 본 실험에서는 식(13)과 같이 오탐률

(False Positive Rate)을 성능 평가 기준에 포함한다.

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \quad (13)$$

제 2 절 실험 내용 및 결과

1. 실험 데이터 분석 및 전처리

본 실험에서 사용한 CICIDS2017 데이터셋은 네트워크 트래픽을 캡처한 PCAP(Packet Capture) 파일 형태로 제공된다. 이 파일을 네트워크 트래픽 플로우 분석 툴인 CICFlowMeter[35, 36]를 이용하여 정형화된 양방향 네트워크 플로우 데이터로 변환 후 사용했다. 변환된 데이터의 예시는 [표 4 -3]과 같다.

[표 4 -3] CICIDS2017 트래픽 플로우 데이터 예시 [30]

Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Label
192.168.10.3-192.168.10.50-3268-56108-6	192.168.10.50	56108	192.168.10.3	3268	6	7/7/2017 8:59	112740690	BENIGN
192.168.10.3-192.168.10.50-389-42144-6	192.168.10.50	42144	192.168.10.3	389	6	7/7/2017 8:59	112740560	BENIGN
8.0.6.4-8.6.0.1-0-0-0	8.6.0.1	0	8.0.6.4	0	0	7/7/2017 9:00	113757377	BENIGN
192.168.10.9-224.0.0.252-63210-5355-17	192.168.10.9	63210	224.0.0.252	5355	17	7/7/2017 9:00	100126	BENIGN
192.168.10.9-224.0.0.22-0-0-0	192.168.10.9	0	224.0.0.22	0	0	7/7/2017 9:00	54760	BENIGN

변환된 데이터는 플로우마다 FlowID, SourceIP, SourcePort, DestinationIP, DestinationPort, Protocol 등 [표 4 -4]와 같이 총 84개의 특징(Feature)으로 이루어져 있으며, 주요 특징 정보는 [표 4 -5]에서 확인할 수 있다.

[표 4-4] CICIDS2017 특징 정보 [30]

번호	특징	번호	특징	번호	특징
1	FlowID	29	FwdIATStd	57	ECEFlagCount
2	SourceIP	30	FwdIATMax	58	Down/UpRatio
3	SourcePort	31	FwdIATMin	59	AveragePacketSize
4	DestinationIP	32	BwdIATTotal	60	AvgFwdSegmentSize
5	DestinationPort	33	BwdIATMean	61	AvgBwdSegmentSize
6	Protocol	34	BwdIATStd	62	FwdAvgBytes/Bulk
7	Timestamp	35	BwdIATMax	63	FwdAvgPackets/Bulk
8	FlowDuration	36	BwdIATMin	64	FwdAvgBulkRate
9	TotalFwdPackets	37	FwdPSHFlags	65	BwdAvgBytes/Bulk
10	TotalBackwardPackets	38	BwdPSHFlags	66	BwdAvgPackets/Bulk
11	TotalLengthofFwdPackets	39	FwdURGFlags	67	BwdAvgBulkRate
12	TotalLengthofBwdPackets	40	BwdURGFlags	68	SubflowFwdPackets
13	FwdPacketLengthMax	41	FwdHeaderLength	69	SubflowFwdBytes
14	FwdPacketLengthMin	42	BwdHeaderLength	70	SubflowBwdPackets
15	FwdPacketLengthMean	43	FwdPackets/s	71	SubflowBwdBytes
16	FwdPacketLengthStd	44	BwdPackets/s	72	Init_Win_bytes_forward
17	BwdPacketLengthMax	45	MinPacketLength	73	Init_Win_bytes_backward
18	BwdPacketLengthMin	46	MaxPacketLength	74	Act_data_pkt_fwd
19	BwdPacketLengthMean	47	PacketLengthMean	75	min_seg_size_forward
20	BwdPacketLengthStd	48	PacketLengthStd	76	ActiveMean
21	FlowBytes/s	49	PacketLengthVariance	77	ActiveStd
22	FlowPackets/s	50	FINFlagCount	78	ActiveMax
23	FlowIATMean	51	SYNFlagCount	79	ActiveMin
24	FlowIATStd	52	RSTFlagCount	80	IdleMean
25	FlowIATMax	53	PSHFlagCount	81	IdleStd
26	FlowIATMin	54	ACKFlagCount	82	IdleMax
27	FwdIATTotal	55	URGFlagCount	83	IdleMin
28	FwdIATMean	56	CWEFlagCount	84	Label

[표 4 -5] CICIDS2017 주요 특징 정보 [30]

번호	특징	설명
8	Flow Duration	네트워크 플로우의 전체 소요 시간
21	Flow Bytes/s	1초당 플로우 바이트 수
22	Flow Packets/s	1초당 플로우 패킷 수
23	Flow IAT Mean	두 패킷 사이의 양방향 도착 간격 시간(Inter-arrival Time)의 평균
28	Fwd IAT Mean	두 패킷 사이의 정방향 도착 간격 시간(Inter-arrival Time)의 평균
33	Bwd IAT Mean	두 패킷 사이의 역방향 도착 간격 시간(Inter-arrival Time)의 평균
76	Active Mean	네트워크 플로우가 활성화된 시간의 평균
80	Idle Mean	네트워크 플로우가 대기 상태인 시간의 평균

본 실험에서는 FlowID와 SourceIP 등 특정 네트워크 환경을 나타내는 특징을 제거하고 레이블 포함 총 77개의 특징 정보를 바탕으로 모델을 학습시킨다.

CICIDS2017 데이터셋은 정상(Benign) 트래픽과 DoS, DDoS, Heartbleed, Web Attack, Infiltration 등 14개 유형의 공격 트래픽으로 구성되어있다. 트래픽 유형별 샘플 수는 [표 4 -6], [표 4 -7]과 같다.

[표 4 -6] CICIDS2017 정상 트래픽 샘플 수 [30]

유형	세부 유형	샘플 수	총 샘플 수
Benign	Monday-WorkingHours	529,918	2,273,097
	Tuesday-WorkingHours	432,074	
	Wednesday-WorkingHours	440,031	
	Thursday-WorkingHours	456,752	
	Friday-WorkingHours	414,322	
합 계			2,273,097

[표 4 -7] CICIDS2017 공격 트래픽 유형별 샘플 수 [30]

유형	세부 유형	샘플 수	총 샘플 수
DoS	DoS Hulk	231,073	252,661
	DoS Goldeneye	10,293	
	DoS Slowloris	5,796	
	DoS Slowhttptest	5,499	
Port Scan	Nmap Port Scan	158,930	158,930
DDoS	DDoSLOIT	128,027	128,027
Brute-Force	FTP-Patator	7,938	13,835
	SSH-Patator	5,897	
Web Attack	Web Brute-Force	1,507	2,180
	XSS	652	
	SQL Injection	21	
Botnet	Botnet Ares	1,966	1,966
Infiltration	Infiltration	36	36
Heartbleed	Heartbleech	11	11
합 계			557,646

DoS 공격은 시스템을 악의적으로 공격해 시스템의 자원을 부족하게 하여 원래 의도된 용도로 사용하지 못하게 하는 공격으로, CICIDS2017 데이터셋의 DoS 공격 샘플은 공격에 사용한 툴과 방식에 따라 DoS Hulk^⑧, Dos Goldeneye^⑨, DoS Slowloris^⑩, DoS Slowhttptest^⑪로 구분된다. 본 실험에서는 다양한 세부 유형으로 구분되고 샘플 수도 풍부한 DoS 공격 트래픽을 실험 대상으로 선정한다.

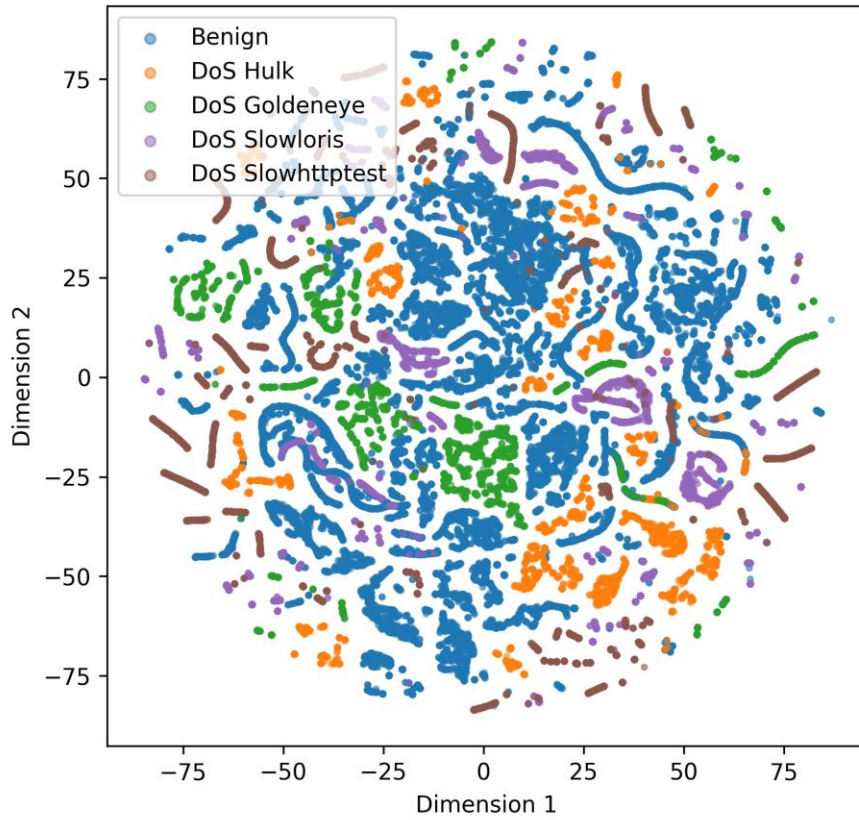
정상 트래픽과 DoS 공격 트래픽 데이터의 분포를 확인하기 위해, 고차원 데이터를 시각화하는 대표적인 방법인 t-SNE (t-Stochastic Neighborhood Embedding)을 통해 2차원의 임베딩 벡터로 변환하여 시각화했다. 실험에 사용한 데이터는 [그림 4 -2]와 같이 높은 비선형 특성을 나타내고 정상 트래픽과 공격 트래픽이 동일한 특성 공간을 공유하므로, 이를 분류하기 위해서는 복잡한 비선형 관계를 모델링할 수 있는 모델이 필요하다. 딥러닝은 고차원의 비선형 데이터를 처리하는 데 우수한 성능을 나타내고 있으므로, 본 연구에서는 딥러닝 모델을 기반으로 하는 네트워크 이상징후 탐지를 진행한다.

^⑧ <https://github.com/grafov/hulk>

^⑨ <https://github.com/jseidl/GoldenEye>

^⑩ <https://github.com/gkbrk/slowloris>

^⑪ <https://github.com/shekyan/slowhttptest>



[그림 4-2] t-SNE를 통한 데이터 시각화

모델을 학습하기에 앞서, 데이터셋의 중복된 데이터와 결측치 데이터를 제거하고, 각 데이터의 값은 아래 식(14)와 같이 최소-최대 정규화를 통해 0과 1 사이의 값으로 변환한다.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (14)$$

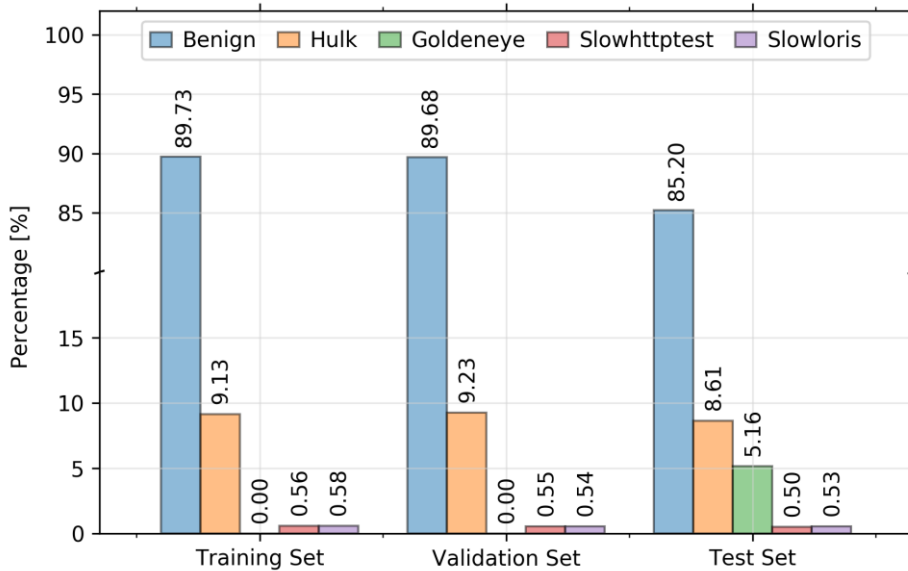
본 실험에서 학습 데이터와 평가 데이터는 8:2 비율로 분리하고, 학습 데이터의 20%는 학습 과정에 대한 검증을 위해 사용한다. 그리고 새로운 공격 유형에 대한 탐지 성능 확인을 위해 DoS Hulk, DoS Goldeneye, DoS Slowloris, DoS Slowhttptest 중 가장 최신 공격 유형이면서 HTTP 플러딩(Flooding)과 CC (Cache Control) 공격이 가능한 DoS Goldeneye 공격을 새로운 공격 유형으로 가정한다. 즉, DoS Goldeneye 공격은 학습 데이터에서는 제외하고 평가 데이터에만 포함

되도록 한다. 최종 데이터의 유형별 샘플 수는 [표 4-8]과 같다.

[표 4-8] 최종 데이터의 유형별 샘플 수

유형	학습 데이터	검증 데이터	평가 데이터
Benign	543,289	135,746	169,967
DoS Hulk	55,272	13,977	17,174
DoS Goldeneye	0	0	10,286
DoS Slowhttptest	3,398	829	1,001
DoS Slowloris	3,505	814	1,066
정상 트래픽	543,289	135,746	169,967
공격 트래픽	62,175	15,620	29,527

학습 데이터, 검증 데이터, 평가 데이터의 각 트래픽 유형별 비율은 [그림 4-3]을 통해 확인할 수 있다.

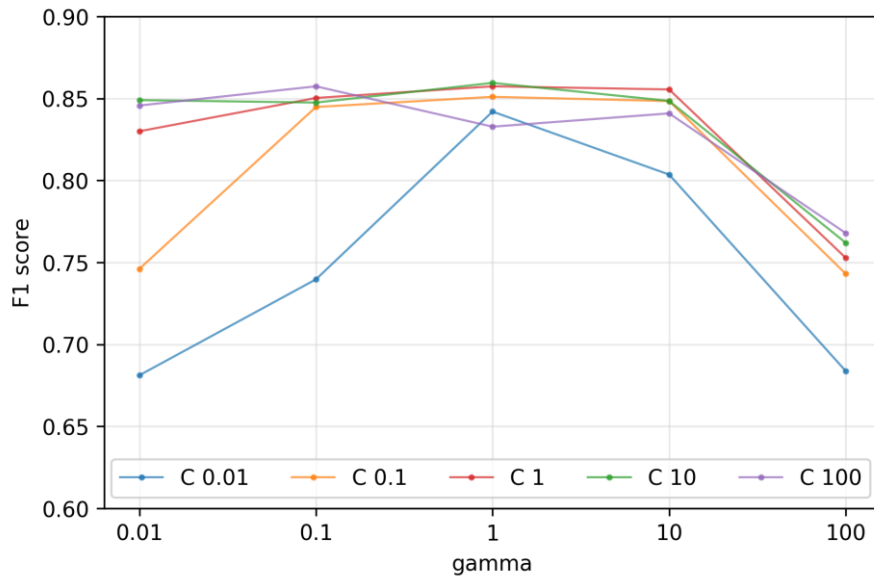


[그림 4-3] 데이터셋의 트래픽 유형별 비율

2. 서포트벡터머신 기반 네트워크 이상징후 탐지 모델

본 실험에서는 가우시안 방사 기저 함수를 커널 함수로 사용한 서포트 벡터머신으로 네트워크 이상징후를 탐지했다. [표 3-2]의 파라미터를

기준으로 그리드 서치(Grid Search)를 통해 C와 gamma의 모든 조합에 대한 탐지 성능을 확인했다.



[그림 4-4] C와 gamma의 변화에 따른 F1 스코어

[그림 4-4]는 서포트벡터머신의 C와 gamma의 변화에 따른 F1 스코어를 보여준다. C 값이 증가할수록 F1 스코어가 높아지다가 C 값이 10에서 가장 높은 성능을 보였고, gamma 값 역시 증가할수록 F1 스코어가 높아지다가 gamma 값이 1에서 최적의 성능을 나타냄을 확인할 수 있었다. [표 4-9]는 F1 스코어 기준 상위 5개의 모델의 성능 측정 결과를 나타낸다.

[표 4-9] 서포트벡터머신 모델의 탐지 성능

파라미터		정확도	정밀도	재현율	F1스코어	오탐률
C	gamma					
10	1	0.9633	0.9901	0.7594	0.8595	0.0013
1	1	0.9629	0.9960	0.7527	0.8574	0.0005
100	0.1	0.9628	0.9922	0.7548	0.8574	0.0010
1	10	0.9624	0.9905	0.7529	0.8555	0.0013
0.1	1	0.9615	0.9961	0.7428	0.8510	0.0005

서포트벡터머신 모델의 새로운 공격 유형에 대한 탐지 성능을 확인했다. 탐지 성능은 새로운 공격 유형으로 가정한 DoS Goldeneye 공격 트래픽을 실제 공격으로 탐지한 비율로 측정했다. 측정 결과는 [표 4-10]과 같다.

[표 4-10] 서포트벡터머신 모델의 새로운 공격 탐지율

파라미터		DoS Goldeneye 탐지율
C	gamma	
10	1	0.4392
1	1	0.4611
100	0.1	0.4296
1	10	0.4215
0.1	1	0.4673

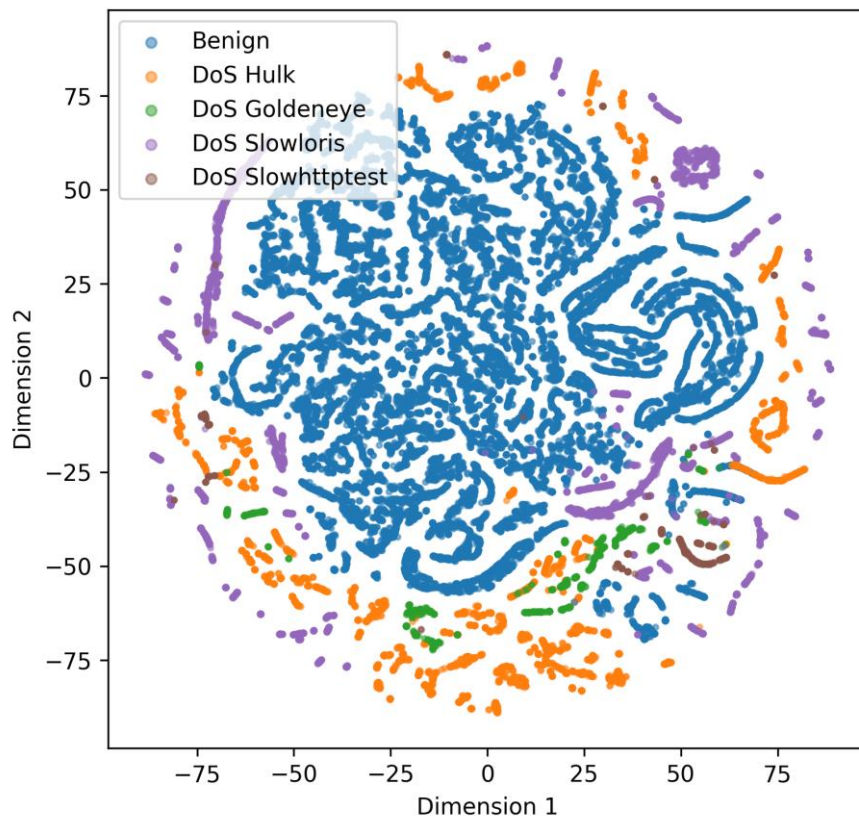
3. 심층신경망 기반 네트워크 이상징후 탐지 모델

본 실험에서는 정상 트래픽 데이터와 공격 트래픽 데이터를 모두 이용하여 학습하는 심층신경망으로 네트워크 이상징후를 탐지했다. 심층신경망의 최적 모델을 탐색하기 위해서 하이퍼 파라미터를 변경해가면서 실험을 수행했다. 은닉층의 개수에 따라 두 개인 경우 유형1, 세 개인 경우 유형2, 네 개인 경우 유형3, 다섯 개인 경우 유형4로 총 네 가지의 모델을 생성하고, 나머지 하이퍼 파라미터는 [표 3-3]을 기준으로 설정했다. 은닉층의 뉴런 수를 4부터 128까지 배수로 증가시키면서 각각 성능을 측정했다. 모델별 최적의 뉴런 구성과 성능 측정 결과는 [표 4-11]과 같다.

[표 4-11] 심층신경망 모델의 탐지 성능

모델	은닉층 구조	정확도	정밀도	재현율	F1스코어	오탐률
유형1	[16, 4]	0.9721	0.9791	0.8295	0.8981	0.0031
유형2	[128, 32, 8]	0.9671	0.9834	0.7907	0.8766	0.0023
유형3	[128, 64, 8, 4]	0.9685	0.9896	0.7955	0.8820	0.0015
유형4	[64, 32, 16, 8, 4]	0.9723	0.9863	0.8242	0.8980	0.0020

본 실험에서 유형1 모델이 최적의 성능을 나타냈고, 유형1 모델의 F1 스코어는 0.8981로 측정되었다. 유형1 모델의 학습 상태를 직관적으로 확인하기 위해, 마지막 완전연결계층에서의 출력 데이터의 분포를 t-SNE를 이용하여 [그림 4-5]와 같이 시각화했다. 입력층에서의 트래픽 유형별 패턴이 [그림 4-2]와 같이 서로 뒤섞인 것에 비하여 모델 심층으로 갈수록 정상 트래픽과 공격 트래픽의 패턴이 구분됨을 확인할 수 있었다.



[그림 4-5] 심층신경망 모델 통과 후 데이터 시각화

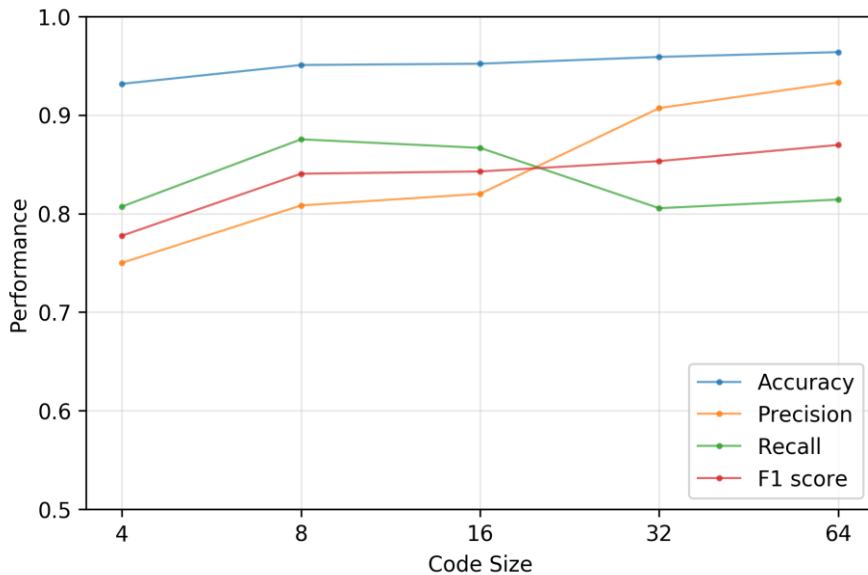
심층신경망 모델의 새로운 공격 유형에 대한 탐지 성능은 [표 4-12]와 같이 확인되었다. 네 가지 유형 중 F1 스코어가 가장 높게 측정된 유형1 모델에서 DoS Goldeneye 탐지율도 가장 높았다.

[표 4-12] 심층신경망 모델의 새로운 공격 탐지율

모델	은닉층 구조	DoS Goldeneye 탐지율
유형1	[16, 4]	0.5143
유형2	[128, 32, 8]	0.4022
유형3	[128, 64, 8, 4]	0.4284
유형4	[64, 32, 16, 8, 4]	0.5053

4. 오토인코더 기반 네트워크 이상징후 탐지 모델

본 실험에서는 정상 트래픽 데이터만을 이용하여 학습하는 오토인코더로 네트워크 이상징후를 탐지했다. 기본 형태의 오토인코더와 잡음 제거 오토인코더, 희소 오토인코더, 적층 오토인코더, 적층 희소 오토인코더 모델을 생성하여 각각 성능을 측정했다. 오토인코더의 최적 모델을 탐색하기 위해서 [표 3-4]를 기준으로 하이퍼 파라미터를 변경해가면서 실험을 수행했다.

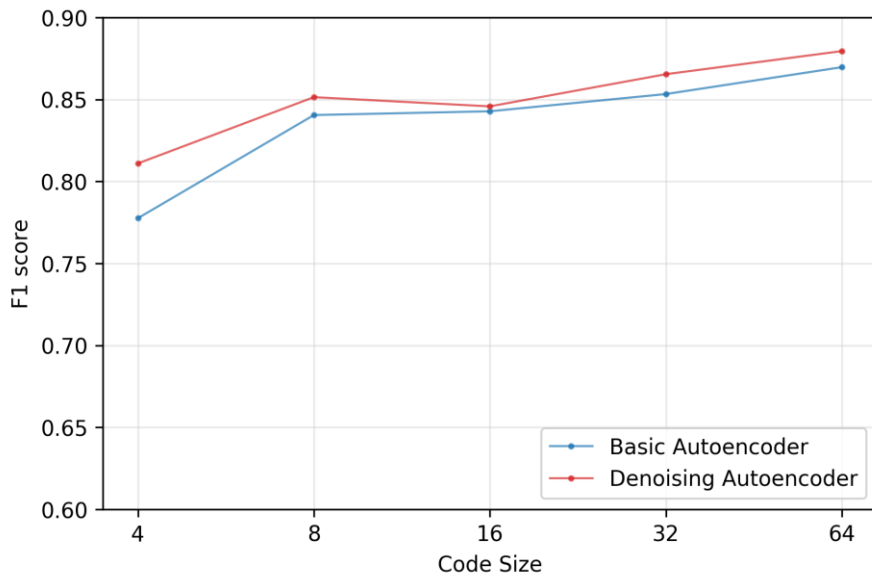


[그림 4-6] 기본 오토인코더 모델의 코드 크기별 성능

먼저, 오토인코더의 코드 크기를 4부터 64까지 배수로 증가시키면서 탐지 성능을 측정했다. [그림 4-6]과 같이 코드 크기가 64일 때 F1 스코어가 가장 높게 측정되었다.

[표 4-13] 기본 오토인코더 모델의 탐지 성능

코드 크기	정확도	정밀도	재현율	F1스코어	오탐률
4	0.9317	0.7501	0.8072	0.7776	0.0467
8	0.9508	0.8083	0.8754	0.8405	0.0361
16	0.9521	0.8201	0.8667	0.8428	0.0330
32	0.9590	0.9071	0.8055	0.8533	0.0143
64	0.9639	0.9331	0.8143	0.8697	0.0101



[그림 4-7] 잡음제거 오토인코더 모델의 코드 크기별 성능

다음으로 잡음제거 오토인코더 모델의 탐지 성능을 확인했다. 입력 데이터에 가우시안 노이즈를 추가하고, 오토인코더가 원본 입력 데이터를 복원하도록 학습시켰다. 잡음제거 오토인코더의 성능이 기본 형태의 오토인코더보다 향상됨을 [그림 4-7]과 같이 확인할 수 있었다.

[표 4-14] 잡음제거 오토인코더 모델의 탐지 성능

코드 크기	정확도	정밀도	재현율	F1스코어	오탐률
4	0.9421	0.7850	0.8387	0.8110	0.0399
8	0.9581	0.8960	0.8110	0.8514	0.0163
16	0.9538	0.8361	0.8556	0.8458	0.0291
32	0.9604	0.8706	0.8602	0.8654	0.0222
64	0.9651	0.8993	0.8606	0.8795	0.0167

희소 오토인코더 모델은 코드 크기가 입력 데이터의 차원보다 크더라도 의미 있는 정보를 학습할 수 있도록 코드층의 출력값에 L1 규제(L1 Regularization)를 추가했다. 희소 오토인코더의 탐지 성능은 [표 4-15]와 같고, 잡음제거 오토인코더보다 높은 탐지 성능을 보였다.

[표 4-15] 희소 오토인코더 모델의 탐지 성능

코드 크기	정확도	정밀도	재현율	F1스코어	오탐률
128	0.9661	0.8921	0.8771	0.8845	0.0184
256	0.9651	0.9273	0.8294	0.8756	0.0113

적층 오토인코더 모델은 코드 크기가 8일 때 가장 높은 성능을 나타냈다. 인코더의 은닉층은 1개부터 3개까지 늘려가면서 측정하였고, 디코더의 은닉층은 인코더의 은닉층과 대칭 구조를 갖도록 했다. 적층 오토인코더 모델의 탐지 성능은 [표 4-16]과 같다.

[표 4-16] 적층 오토인코더 모델의 탐지 성능

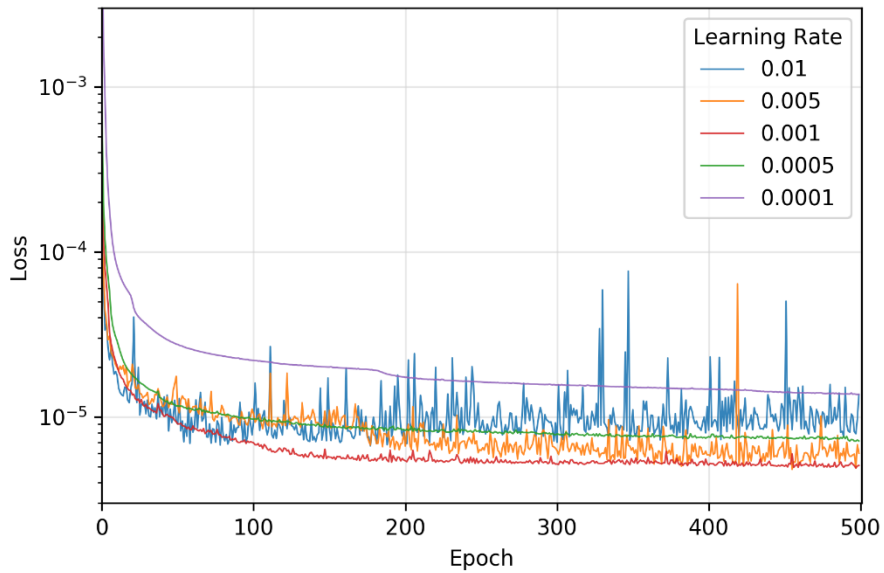
인코더 구조	코드 크기	정확도	정밀도	재현율	F1스코어	오탐률
[16]	8	0.9607	0.8989	0.8278	0.8619	0.0162
[32, 16]	8	0.9696	0.9388	0.8498	0.8921	0.0096
[64, 32, 16]	8	0.9752	0.9483	0.8806	0.9132	0.0083

적층 희소 오토인코더 모델은 인코더의 은닉층 구조가 [128, 32, 16]이고 코드 크기가 8일 때 F1 스코어가 0.9163으로 가장 높은 성능을 나타냈다. [표 4-17]은 F1 스코어 기준 상위 세 개 모델의 탐지 성능을 보여준다.

[표 4-17] 적층 희소 오토인코더 모델의 탐지 성능

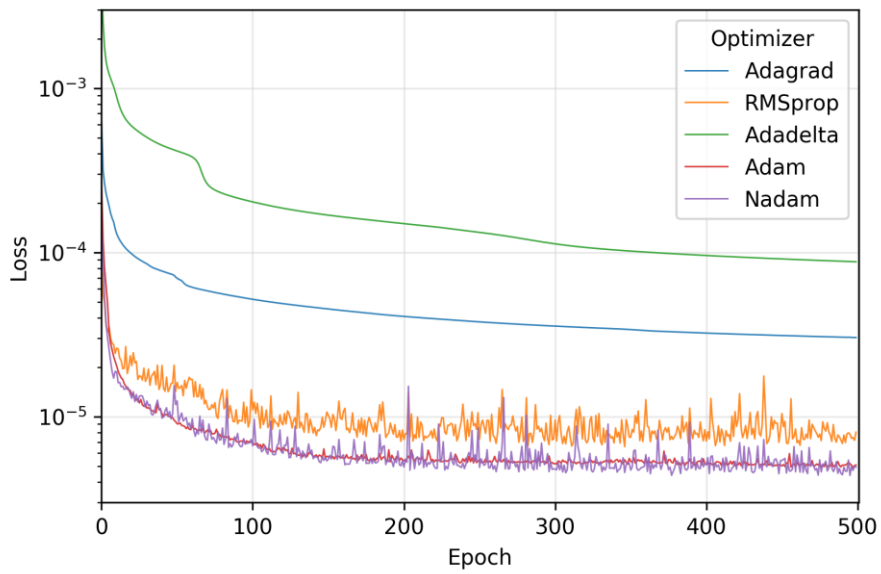
인코더 구조	코드 크기	정확도	정밀도	재현율	F1스코어	오탐률
[128, 32, 16]	8	0.9758	0.9391	0.8947	0.9163	0.0101
[128, 64]	8	0.9746	0.9383	0.8865	0.9117	0.0101
[128, 16]	8	0.9726	0.9215	0.8909	0.9059	0.0132

오토인코더 모델의 학습 과정에서 학습률(Learning Rate)의 영향을 확인하기 위해 학습률을 변경해가면서 손실 감소를 확인했다. 학습률을 0.001로 설정했을 때 가장 안정적인 손실 감소를 나타냈다. [그림 4-8]에서 학습률에 따른 손실 변화를 확인할 수 있다.



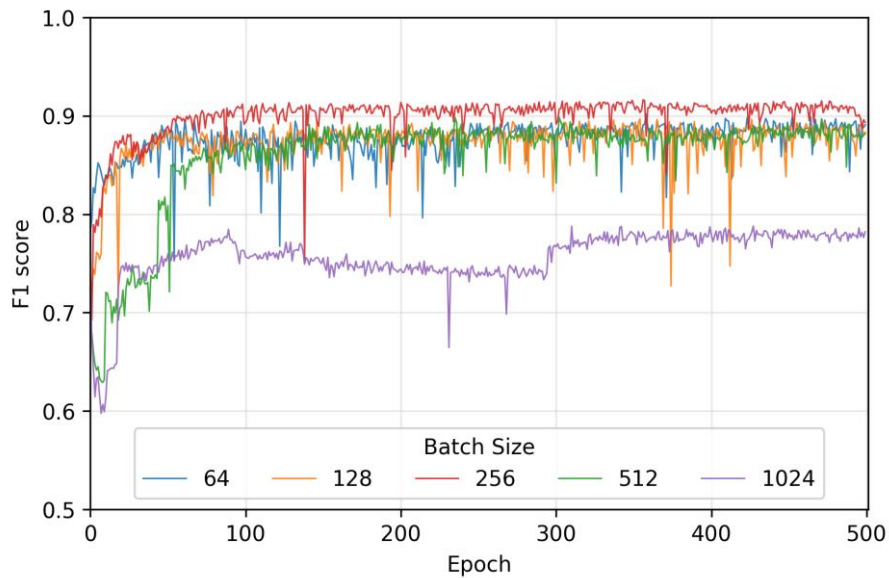
[그림 4-8] 학습률에 따른 손실 변화

옵티마이저는 Adam을 사용했을 때 가장 안정적인 손실 감소를 나타냈으며, [그림 4-9]에서 옵티마이저에 따른 손실 변화를 확인할 수 있다.



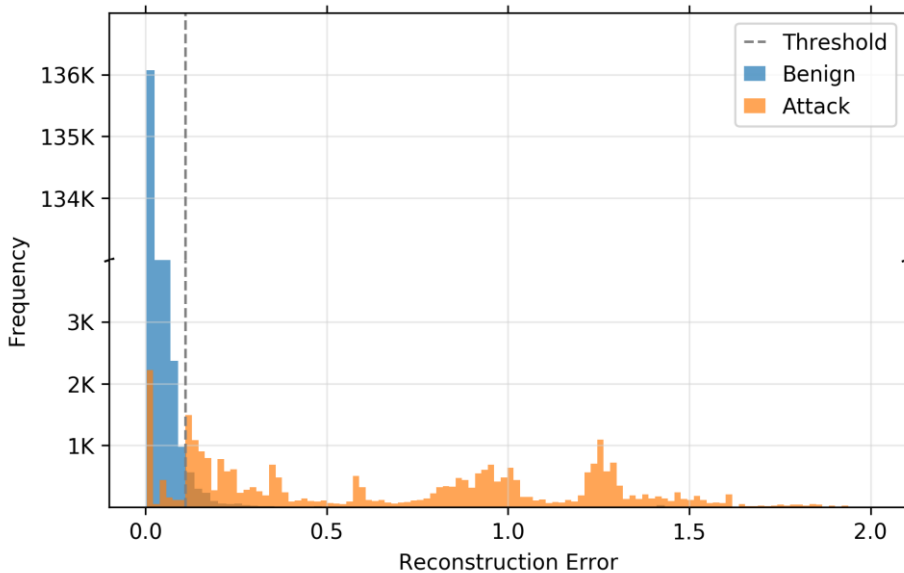
[그림 4-9] 옵티마이저에 따른 손실 변화

배치 크기(Batch Size)는 256으로 설정 시 가장 높은 성능을 나타냈고, [그림 4-10]에서 배치 크기에 따른 탐지 성능을 확인할 수 있다.



[그림 4-10] 배치 크기에 따른 성능

오토인코더 모델 중 가장 높은 탐지 성능을 나타낸 적층 희소 오토인코더에서 정상 트래픽과 공격 트래픽의 복원 오차의 분포는 [그림 4-11]과 같이 나타났다. 정상 트래픽의 복원 오차는 0 주변에 분포하고 있는 반면, 공격 트래픽의 복원 오차는 넓은 범위에 분포하고 있다.

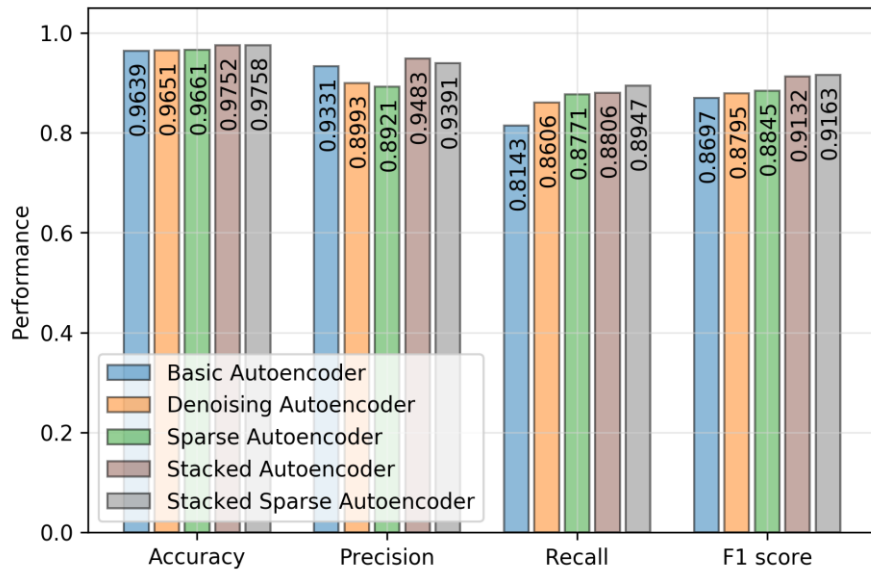


[그림 4-11] 적층 희소 오토인코더 모델의 복원 오차 분포

지금까지 살펴본 오토인코더 모델별 탐지 성능을 정리하면 [표 4-18]과 같다. F1 스코어 기준으로 적층 희소 오토인코더 모델에서 가장 높은 탐지 성능을 나타냈지만, 오탐률은 적층 오토인코더 모델에서 가장 낮게 측정되었다.

[표 4-18] 오토인코더 모델별 탐지 성능

모델	정확도	정밀도	재현율	F1스코어	오탐률
오토인코더	0.9639	0.9331	0.8143	0.8697	0.0101
잡음 제거 오토인코더	0.9651	0.8993	0.8606	0.8795	0.0167
희소 오토인코더	0.9661	0.8921	0.8771	0.8845	0.0184
적층 오토인코더	0.9752	0.9483	0.8806	0.9132	0.0083
적층 희소 오토인코더	0.9758	0.9391	0.8947	0.9163	0.0101



[그림 4-12] 오토인코더 모델별 탐지 성능

오토인코더 모델의 새로운 공격 유형에 대한 탐지 성능은 [표 4-19]와 같이 확인되었다. 적층 희소 오토인코더 모델에서 DoS Goldeneye 탐지율이 0.9111로 가장 높았다.

[표 4-19] 오토인코더 모델별 새로운 공격 탐지율

모델	DoS Goldeneye 탐지율
오토인코더	0.7336
잡음 제거 오토인코더	0.8644
희소 오토인코더	0.8484
적층 오토인코더	0.8658
적층 희소 오토인코더	0.9111

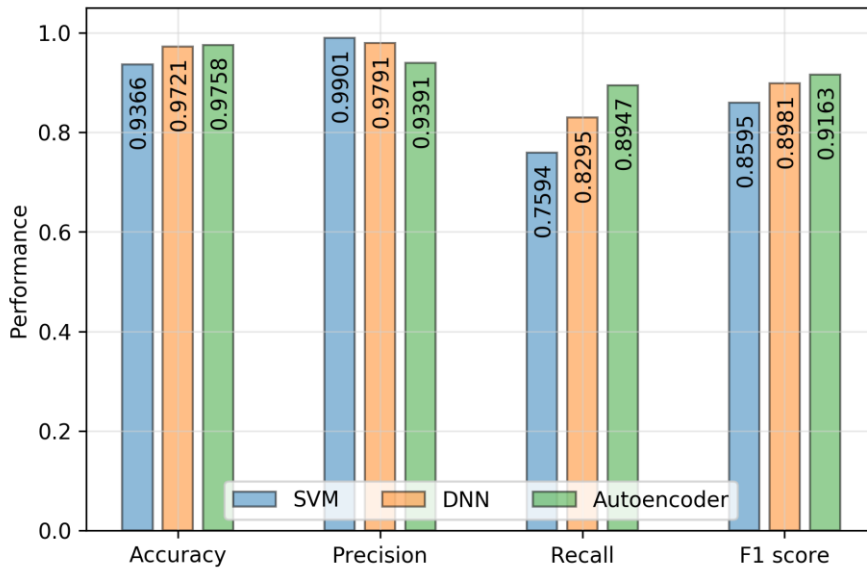
5. 성능 평가

본 실험에서는 서포트벡터머신 모델과 심층신경망 모델, 오토인코더 모델의 네트워크 이상징후 탐지 성능을 확인하였다. 서포트벡터머신 모델은 C 10, gamma 1에서 최적의 성능을 나타냈고, 심층신경망 모델은 두 개의 은닉층으로 구성된 모델에서 최적의 성능을 나타냈다. 또한 오토인코더 모델은 적층 희소 오토인코더 모델에서 최적의 성능을 나타냈

다. 탐지 모델별 성능은 [표 4-20]에서 확인할 수 있다. 오토인코더 모델의 F1 스코어는 0.9163으로 서포트벡터머신 모델이나 심층신경망 모델보다 높게 나타났다.

[표 4-20] 네트워크 이상징후 탐지 모델의 성능 비교

모델	정확도	정밀도	재현율	F1스코어	오탐률
서포트벡터머신	0.9633	0.9901	0.7594	0.8595	0.0013
심층신경망	0.9721	0.9791	0.8295	0.8981	0.0031
오토인코더	0.9758	0.9391	0.8947	0.9163	0.0101

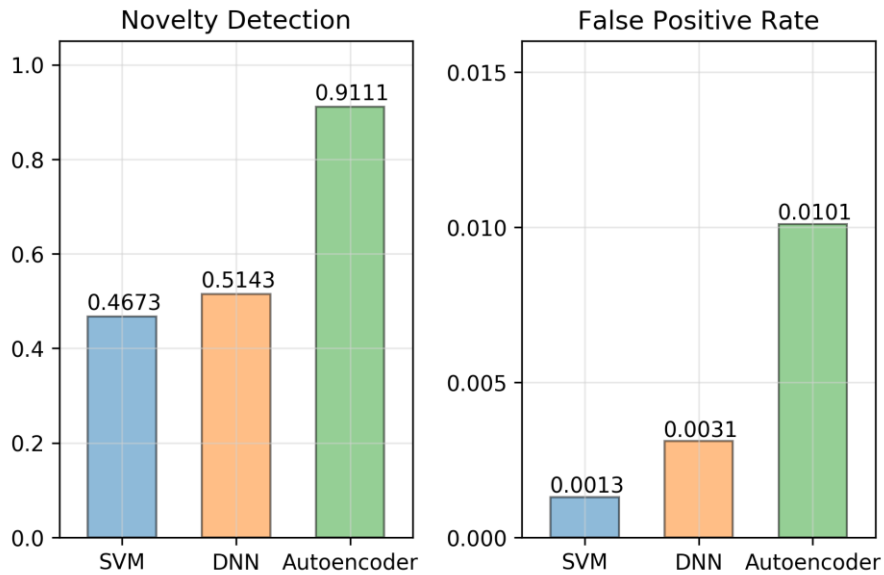


[그림 4-13] 네트워크 이상징후 탐지 모델의 성능 비교

또한, 각 탐지 모델이 새로운 공격 유형으로 설정한 DoS Goldeneye에 대해 공격으로 탐지해낸 비율을 [그림 4-14]를 통해 확인해보면 오토인코더 모델의 탐지율이 0.9111로 서포트벡터머신 모델이나 심층신경망 모델보다 월등히 높게 나타났다. 이를 통해 기존에 알려지지 않은 새로운 공격 트래픽을 탐지함에 있어서 오토인코더 모델이 적합함을 확인할 수 있었다.

하지만 정상 트래픽을 공격 트래픽으로 잘못 판단하는 오탐률 측면에서는 오토인코더 모델의 성능이 다른 두 모델보다 조금 떨어졌다. 그 이유는 오토인코더 모델에서 정상 트래픽과 공격 트래픽을 구분하는 임계

치의 최적값을 F1 스코어를 기준으로 설정함으로 인해 일부 정상 트래픽의 복원 오차가 임계치보다 커지게 되고, 이를 공격 트래픽으로 잘못 판단하기 때문이다. 그러나 각 모델별 오탐률의 차이는 F1 스코어의 차이에 비해 상대적으로 무시할 수 있는 정도로 나타났다.



[그림 4-14] 탐지 모델별 새로운 공격 탐지율과 오탐률

제 5 장 결 론

제 1 절 연구 성과

본 연구에서는 자기지도학습 기반의 오토인코더를 이용한 네트워크 트래픽 이상징후 탐지 모델을 제안하였다. 그리고 최근에 수집된 실제 네트워크 트래픽 데이터셋을 사용하여 제안한 모델의 탐지 성능을 확인하였다. 또한 기존에 알려지지 않은 새로운 공격에 대한 탐지 상황을 설정하고 그에 대한 모델의 탐지 성능을 확인하였다.

본 연구에서 제안한 모델의 효과를 검증하기 위해, 지도학습 기반의 서포트벡터머신 모델, 심층신경망 모델과 탐지 결과를 비교했다. 네트워크 트래픽 데이터는 그 특성상 데이터셋의 클래스 불균형이 발생할 수 있기 때문에 정확도만으로 모델의 성능을 판단하기에는 한계가 있다. 그래서 정밀도와 재현율의 조화 평균인 F1 스코어를 기준으로 모델의 성능을 측정하고 기존 탐지 모델의 성능과 비교했다.

본 연구에서는 기본 형태의 오토인코더 모델뿐만 아니라 잡음제거 오토인코더, 희소 오토인코더, 적층 오토인코더, 적층 희소 오토인코더 모델의 탐지 성능을 측정하고, 네트워크 이상징후 탐지에 가장 적합한 모델을 찾았다. 실험을 통해서 각 오토인코더 모델의 최적화된 구조를 찾고 성능을 측정한 결과, 적층 희소 오토인코더 모델이 정확도 0.9758, F1 스코어 0.9163으로 가장 우수한 성능을 나타냈다. 오토인코더 모델은 서포트벡터머신 모델보다 F1 스코어 기준 5%p 이상 성능이 향상되었고, 새로운 공격 유형에 대해서도 기존 모델보다 40%p 가량 높게 탐지해내는 것을 확인할 수 있었다.

기존에 이루어진 대다수의 네트워크 이상징후 탐지 연구는 지도학습을 기반으로 하고 있다. 하지만 지도학습 기반의 탐지 모델은 학습에 사용할 데이터를 정상 트래픽과 공격 트래픽으로 레이블링해야 한다는 한계가 있었다. 그에 반해, 본 연구에서 제안한 자기지도학습 기반의 오토인코더는 공격 트래픽 데이터 없이도 정상 트래픽 데이터만으로 모델을 학습할 수 있다는 장점이 있었다. 게다가 본 연구에서는 최근에 수집된 트래픽 데이터셋을 사용하여 최신 네트워크 침입 패턴과 실제 네트워크 환경을 반영하고 실험 결과에 대한 타당성을 높일 수 있었다.

본 연구에서 제안한 자기지도학습 기반의 오토인코더를 이용한 네트

워크 이상징후 탐지 모델은 네트워크 침입탐지시스템의 성능 향상에 기여할 수 있을 것으로 기대한다.

제 2 절 보완 사항 및 향후 계획

본 연구에서는 샘플 수가 충분하고 세부 공격 유형으로 분리 가능한 DoS 공격 트래픽을 실험의 대상으로 선정했다. 이를 통해 새로운 공격 유형에 대한 탐지 상황을 설정하고 그에 대한 모델의 탐지 성능을 확인할 수 있었다. 하지만 DoS 공격만을 대상으로 한정시키다 보니 다양한 네트워크 공격 유형을 본 실험에 반영하지는 못했다. 따라서 향후 연구에서는 다양한 공격 유형으로 구성되어있고 각각 충분한 샘플 수를 가진 네트워크 트래픽 데이터셋을 확보하여 추가 실험을 진행하고자 한다.

또한 본 연구에서 제안한 지도학습 기반의 오토인코더 모델은 F1 스코어 기준 가장 우수한 성능을 나타냈지만, 정밀도는 다른 두 모델보다 낮게 측정되었고 오탐률 역시 심층신경망 모델보다 높았다. 네트워크 침입탐지시스템에서 오탐이 증가하면 전체 로그의 발생량이 증가하게 되고 실제 공격을 탐지하기 어려울 수 있다. 따라서 오탐률을 줄일 수 있는 방안에 대한 추가 연구가 필요하다. 최근 이상징후 탐지 분야에서 딥러닝 기반 생성 모델 중 하나인 생성적 적대 신경망(Generative Adversarial Network, GAN)이 좋은 성과를 내고 있다. 네트워크 이상징후 탐지 모델에도 생성적 적대 신경망 모델을 이용하여 트래픽 데이터의 분포를 학습하고, 이상 점수(Anomaly Score) 차이를 통해 네트워크 침입을 탐지할 수 있다. 향후 연구에서는 생성적 적대 신경망을 이용하여 네트워크 이상징후를 탐지하는 연구를 통해 네트워크 침입탐지시스템의 성능을 향상시키고자 한다.

참고문헌

- [1] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022," 2018.
- [2] Cisco, "Cisco 2018 Annual Cybersecurity Report," 2018.
- [3] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980.
- [4] N. Pappas, "Network IDS & IPS Deployment Strategies," *SANS Institute*, Apr. 2008.
- [5] M. Thottan and C. Ji, "Anomaly Detection in IP Networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.
- [6] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. LISA '99: 13th Systems Administration Conference*, Nov. 1999, pp. 229–238.
- [7] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.
- [8] D. S. Kim, H. –N. Nguyen, and J. S. Park, "Genetic algorithm to improve SVM based network intrusion detection system," in *19th Int. Conf. Advanced Information Networking and Applications*, March 2005, vol. 1: IEEE, pp. 155–158.
- [9] K. A. Jalil, M. H. Kamarudin, and M. N. Masrek, "Comparison of Machine Learning algorithms performance in detecting network intrusion," in *2010 Int. Conf. Networking and Information Technology*, Jun. 2010: IEEE, pp. 221–226.
- [10] Y. –X. Meng, "The practice on using machine learning for network anomaly intrusion detection," in *2011 Int. Conf. Machine Learning and Cybernetics*, Jul. 2011, vol. 2: IEEE, pp. 576–581.
- [11] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.
- [12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, no. 3, p. 1, 1988.
- [13] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proc. 27th Int. Conf. Machine Learning*, 2010, pp. 807–814.
- [14] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *arXiv:1710.05941v2*, 2017.

- [15] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [16] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," in *Proc. 25th Int. Conf. Machine Learning*, 2008: ACM, pp. 1096–1103.
- [17] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, no. 2011, pp. 1–19, 2011.
- [18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015/05/01 2015, doi: 10.1038/nature14539.
- [19] R. C. Staudemeyer and C. W. Omlin, "Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data," in *Proc. South African Institute for Computer Scientists and Information Technologists Conf.*, Oct. 2013: ACM, pp. 218–224.
- [20] "KDD Cup 1999 Data." The UCI KDD Archive. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (accessed Nov. 19, 2019).
- [21] N. Gao, L. Gao, Q. Gao, and H. Wang, "An Intrusion Detection Model Based on Deep Belief Networks," in *2014 2nd Int. Conf. Advanced Cloud and Big Data*, Nov. 2014: IEEE, pp. 247–252.
- [22] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of Intrusion Detection using Deep Neural Network," in *2017 IEEE Int. Conf. Big Data and Smart Computing*, Feb. 2017: IEEE, pp. 313–316.
- [23] T.-T.-H. Le, J. Kim, and H. Kim, "An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization," in *2017 Int. Conf. Platform Technology and Service*, Feb. 2017: IEEE, pp. 1–6.
- [24] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.
- [25] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in *Proc. 9th EAI Int. Conf. Bio-inspired Information and Communications Technologies*, May 2016: ICST (Institute for Computer Sciences, Social-Informatics and ...), pp. 21–26.
- [26] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symp. Computational Intelligence for Security and Defense*

- Applications*, Jul. 2009: IEEE, pp. 1–6.
- [27] M. Yousefi–Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder–based Feature Learning for Cyber Security Applications," in *2017 Int. Joint Conf. Neural Networks*, May 2017, pp. 3854–3861.
 - [28] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion Detection Using Convolutional Neural Networks for Representation Learning," in *Int. Conf. Neural Information Processing*, Nov. 2017: Springer, pp. 858–866.
 - [29] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al–Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
 - [30] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th Int. Conf. Information Systems Security and Privacy*, 2018, pp. 108–116.
 - [31] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2018, no. 1, pp. 177–200, 2018.
 - [32] C. Cortes and V. Vapnik, "Support–Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
 - [33] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer science & business media, 2013.
 - [34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
 - [35] G. Draper–Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic using Time–related Features," in *Proc. 2nd Int. Conf. Information Systems Security and Privacy*, Feb. 2016, pp. 407–414.
 - [36] A. H. Lashkari, G. Draper–Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic using Time based Features," in *3rd Int. Conf. Information Systems Security and Privacy*, 2017, pp. 253–262.

Abstract

Self-supervised Network Anomaly Detection using Autoencoders

Seungsoo Seo

Department of Engineering Practice
Graduate School of Engineering Practice
Seoul National University

With the development of network and communication technologies, the volume of network traffic is rapidly growing. This has led to an increase of cyber threats, and it is necessary to efficiently detect various types of network attacks. Network Intrusion Detection Systems (NIDSs) can be classified into two types: signature detection and anomaly detection. Signature detection is highly effective in detecting known attacks, but is insufficient against unknown or novel attacks. Anomaly detection is able to detect unknown and novel attacks, but produces a high false positive rate.

Recently, deep learning has achieved great success in various areas such as computer vision, speech recognition, healthcare, etc. In the field of network security, many researchers use deep learning methods to improve detection of network anomalies. However, most of the datasets used in previous studies are out of date and may not represent current patterns of network traffic. Some of these datasets do not cover the variety of attack and are unreliable to use. Moreover, most NIDSs based on supervised learning require data labeling and

may suffer from data imbalance problem.

In this study, we propose a self-supervised network anomaly detection method using autoencoders. Our proposed model uses only benign data for training. In addition, we study the performance of the proposed model on an up-to-date dataset, named CICIDS2017. We compare the performance of the proposed model with those of a conventional support vector machine model and a deep neural network model. The experimental results show that the proposed model outperforms the two baseline models in terms of accuracy, F1-score and novelty detection rate. We believe that these findings will contribute to the development of NIDS.

Keywords: NIDS, Anomaly Detection, Deep Learning, DNN,
Autoencoder

Student Number: 2018-21440