Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.5, Oct. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.5.933

Opcode 빈도수 기반 악성코드 이미지를 활용한 CNN 기반 악성코드 탐지 기법*

고 석 민,[1†] 양 재 혁,[2] 최 원 준,[2] 김 태 근[3‡]
[1,2,3]순천향대학교 (대학원생, 학생, 교수)

CNN-Based Malware Detection Using Opcode Frequency-Based Image*

Seok Min Ko,[1†] JaeHyeok Yang,[2] WonJun Choi,[2] TaeGuen Kim[3‡]
[1,2,3]SoonChunHyang University (Graduate, Undergraduate, Professor)

요 약

인터넷이 발달하고 컴퓨터 이용률이 높아짐에 따라 악성코드로 인한 위협 또한 함께 증가하고 있다. 매년 발견되는 악성코드의 수는 급격히 증가하여 자동으로 대량의 악성코드를 분석하기 위한 시스템이 필요한 상황이다. 본 논문에서는 딥러닝 알고리즘을 활용한 악성코드 자동 분석 기법을 소개한다. CNN(Convolutional Neural Network)라는 이미지 분류에 활용도가 높은 알고리즘을 이용하여 악성코드의 특징을 이미지화한 데이터를 분석한다. 제안하는 방법은 악성코드의 Semantic한 정보를 탐지에 활용하기 위하여 단순 바이너리 바이트를 기반으로 생성한 이미지가 아닌, 바이너리의 명령어 빈도수를 기반으로 생성한 이미지를 CNN으로 분석한다. 악성코드 10,000개 정상코드 10,000개로 구성된 대량의 데이터 셋을 활용하여 탐지 성능을 확인한 결과, 제안하는 방법은 91%의 정확도로 악성코드를 탐지할 수 있음이 확인되었다.

ABSTRACT

As the Internet develops and the utilization rate of computers increases, the threats posed by malware keep increasing. This leads to the demand for a system to automatically analyzes a large amount of malware. In this paper, an automatic malware analysis technique using a deep learning algorithm is introduced. Our proposed method uses CNN (Convolutional Neural Network) to analyze the malicious features represented as images. To reflect semantic information of malware for detection, our method uses the opcode frequency data of binary for image generation, rather than using bytes of binary. As a result of the experiments using the datasets consisting of 20,000 samples, it was found that the proposed method can detect malicious codes with 91% accuracy.

Keywords: Machine Learning, Convolution Neural Network, Clustering, Malware Detection

I. 서 론

최근 인터넷 뱅킹 등 일반 사용자의 편의를 위한 여러 서비스가 제공되고 있다. 그리고 일반 사용자를 대상으로 한 악성코드도 널리 유행하고 있으며, 이로 인한 피해도 꾸준히 증가하는 추세이다[1]. 악성코드에 대한 위협과 피해를 줄이기 위한 악성코드 분석가들의 수는 늘어나고 있지만, 모든 악성코드를 분석가가 직접 수집하여 정밀하게 분석하는 것은 많은 시간과 비용을 발생시킨다. 따라서 대량의 악성코드

Received(08. 10. 2022), Modified(09. 28 .2022), Accepted(09. 28. 2022)
* 본 과제(결과물)는 2022년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업의 결과입니다.(2021RIS-004)
† 주저자, 20164620@sch.ac.kr
‡ 교신저자, tg.kim@sch.ac.kr(Corresponding author)

를 자동으로 분석할 수 있는 시스템이 필요하다. 대표적인 자동 분석 시스템에는 안티바이러스 시스템이 존재한다. 안티바이러스 시스템은 시그니처 기반 방법을 사용하여 알려진 악성코드를 찾아낸다. 알려지지 않은 악성코드의 경우, 전통적인 시그니처 기반 탐지방식으로는 탐지하기가 어렵고 이를 보완하기 위해 최근 기계학습 혹은 딥러닝 알고리즘을 사용하는 연구가 다수 진행되고 있다.〔2〕

본 논문은 명령어 군집화를 활용한 이미지 생성 및 CNN(Convolutional Neural Network) 기반 악성코드 탐지 방법을 제안한다. 기존의 CNN 기반 악성코드 탐지 기법들은〔3,4,5,6〕 주로 바이너리의 바이트들을 변환하여 생성한 이미지를 CNN의 입력으로 활용한다. 우리가 제안하는 방법은 명령어의 2-gram 빈도수를 이미지화하고, CNN에 적용한다. 이때 이미지를 구성하는 행과 열의 좌표 인덱스를 설정하기 위해 두 가지 군집화 방법과 군집화 거리를 계산하는 방법을 같이 제안한다.

II. 관련 연구

악성코드를 분석하는 기술은 크게 두 가지로 정적분석과 동적 분석이 존재한다. 정적분석은 악성코드를 실행하지 않고 분석하는 방법을 의미한다. 정적분석으로 추출 가능한 정보로는 문자열 및 파일 크기, 패킹 여부, 해시값, DLL 사용 여부, 컴파일 시간 등이 존재한다. 실행을 시키지 않기 때문에 실행 흐름, 시스템 환경 변화 등의 Semantic 정보를 얻기 힘들다〔7〕. 동적 분석은 악성코드를 실행하면서 분석하는 방법을 의미한다. 악성코드를 실행함으로써 분석할 수 있는 정보는 파일 수정 여부, 레지스트리 키 변경 행위, 네트워크 접근 행위, 악성코드가 사용하는 API 등의 정보를 취득할 수 있다.

2.1 정적분석을 이용한 악성코드 탐지

정적분석을 이용한 탐지는 악성코드를 실행하지 않고 정보를 수집하여 탐지하는 방법이다.

정적분석을 위해 Opcode를 얻는 방법으로 Pintool〔8〕, Capstone〔9〕을 사용하거나 IDA pro를 이용하여 opcode를 추출할 수 있다.

Ahmed 등〔10〕의 연구에서는 악성코드는 8bit 단위로 끊어져 있다는 점을 이용하여 8bit gray scales로 변환하여 이미지를 제작하여 CNN 알고리즘을 이용하여 악성코드 패밀리를 분류하는 연구를 진행했다. 하지만 해당 연구는 악성코드의 바이트를 그대로 반영하여 파일 엔드 시그니처 이후에 임의의 바이트를 추가한 경우 탐지를 하기 어렵다는 단점이 존재한다. 하지만 본 연구는 악성코드의 명령어 간의 관계와 명령어의 순서를 반영하여 이미지를 제작하여 악성코드의 명령어에 집중한다는 장점이 존재한다.

Wang 등〔11〕의 연구에서는 휴리스틱 정보와 PE파일 항목 등 정적분석을 이용하여 얻어 낸 정보를 이용하여 SVM(Support Vector Machine) 알고리즘에 적용하여 패킹 감지 프레임 워크를 제안한다. A. Yewale 등〔21〕의 연구에서는 악성코드의 명령어를 IDA를 이용하여 추출한 뒤, 각 명령어의 횟수를 기반으로 의사결정트리, SVM, 부스팅 알고리즘, Random Forest 알고리즘을 이용하여 악성코드를 탐지하는 연구를 진행했다.

Baldangombo 등〔22〕의 연구에서는 악성코드의 PE 파일을 정적분석하여 PE 헤더, 사용하는 DLL, DLL 함수를 수집하여 정보이득 알고리즘과 차원축소 알고리즘을 이용하여 중요 특징을 파악하여 의사결정 알고리즘, SVM, 베이지안 룰을 적용하여 악성코드를 탐지하는 연구를 진행했다. S. Naval 등〔23〕의 연구에서는 정적분석을 위하여 objdump를 이용하여 opcode를 추출하고, PEInfo를 사용하여 PE 정보에서 27개의 정보를 선택하고 사용하는 DLL, 사용하는 함수를 이용하여 CNN 알고리즘을 이용하여 악성코드를 탐지한다.

Bakhshinejad〔28〕과 Zhang〔29〕는 우리가 제안하는 방법과 유사하게 Byte n-gram 혹은 Opcode n-gram 데이터를 추출하여 CNN 기반 분류 모델의 입력으로 사용하는 탐지 방법을 제안하고 있다. 제안되는 두 기존 연구 모두 입력으로 사용되는 Byte n-gram 혹은 Opcode n-gram를 입력 벡터의 어느 위치에 배치시킬 것인가에 대한 고려가 없다. 본 논문에서 제안하는 방법은 이미지 생성 시 상관도가 높은 Opcode를 유사한 위치에 배치하여 유사 성격의 Opcode가 탐지에 많은 기여를 할 수 있도록 한다.

2.2 동적분석을 이용한 탐지

동적분석을 이용한 탐지는 악성코드를 실행시켜서 얻는 정보를 이용하여 분석하여 악성코드 여부를 확인한다. Zahra〔12〕 등의 연구에서는 가상환경에서

Windows API 함수, 함수의 인자 값, 반환 값, 중요한 DLL API 접근 등을 이용하여 악성 행위를 수집한다. 수집한 악성 행위들을 AdaBoostM1 알고리즘을 이용하여 실제 악성코드 여부를 판단한다. 해당 연구의 문제는 악성코드를 실행할 가상환경 구축의 번거로움과 악성 행위가 실행되기까지 시간을 알 수 없으므로, 임의로 임계치 시간을 설정하여야 하는 단점이 존재한다. 악성코드의 악성 행위가 가상으로 정한 임계치 시간 그 이후에 동작한다면 실질적인 악성 행위 특징을 추출하지 못하기에 정확한 판단이 불가능하다는 단점이 존재한다. 하지만 본 연구는 명령어를 추출하여 악성코드 여부를 판단하기에 임계치 시간과 무관하게 동작이 가능하다는 장점이 있다.

Muhammad[13] 등의 연구에서는 쿠쿠 샌드박스에서 악성코드를 실행시켜 얻은 정보인 레지스트리 키 변조, 파일 수정 및 접근, 악성코드의 API 호출 정보, 네트워크 트래픽 등을 이용하여 기계학습 알고리즘인 Gradient Boosting 알고리즘을 이용하여 악성코드를 탐지한다. 하지만 해당 연구에서 언급했듯이 악성코드의 발달로 인해 악성코드의 실행 환경이 가상환경에서 실행이 되었는지 자체적으로 판단할 수 있다. 이 같은 경우에는 악성코드의 악성 행위 특징을 수집할 수 없다는 단점이 존재한다. 하지만 본 연구는 악성코드의 실행 여부와는 상관없이 명령어 추출이 가능하다는 장점이 있다.

R. S. Pirscoveanu 등[24]의 연구에서는 쿠쿠 샌드박스를 통하여 악성코드를 Window API 기반으로 동적 분석하여 WEKA를 통하여 의사결정트리 알고리즘을 이용하여 악성코드 패밀리를 분류한다. 하지만 해당 연구에서는 의사결정트리 알고리즘의 문제점인 과대 적합 문제가 존재한다. 본 연구에서는 CNN 모델에서 과대 적합의 문제점을 해결할 수 있는 층을 배치하여 과대 적합 문제를 해결했다.

M. Smith 등[25]의 연구에서는 악성코드를 가상환경에서 실행하여 실행 후 1000개의 시스템 호출 함수를 사용하여 악성코드를 탐지한다. 탐지를 위해서 CNN, LSTM(Long Short-Term Memory) 알고리즘, CNN+LSTM, 히스토그램과 랜덤 포레스트를 합쳐 악성코드를 탐지하는 연구를 했다. K. Aoki 등[26]의 연구에서는 HTTP 통신을 분석하여 악성 사이트를 탐지하는 연구를 진행했다. HTTP 통신을 동적 분석하여 접속하는 페이지에서 IP주소, URL 데이터, 다운로드 정보를 이용하여 SVM 알고리즘을 이용하여 악성 사이트를 탐지한다.

III. 제안 방법

본 연구에서 제안하는 방법의 처리 과정은 Fig 1에 나타나 있다. 제안 방법은 크게 5가지의 단계로 구분할 수 있다.

- 바이너리 디스어셈블링 과정
- 명령어 Opcode 시퀀스 데이터 추출 과정
- 명령어 Opcode 2-gram 빈도수 추출 과정
- 2-gram 빈도수 기반 이미지 생성 과정
- CNN 기반 악성코드 탐지 과정

3.1절부터 3.5절까지 제안하는 방법의 각 단계를 상세하게 설명한다. 제안 방법의 4단계 이미지 생성의 경우, 사전에 이미지 좌표 인덱스를 명령어와 각각 매칭해야 한다. 이를 위해 이미지 좌표 설정을 위한 방법이 필요하다. 이에 대한 설명도 3.4절에 포함되어있다. 다만, 해당 이미지 좌표 설정은 학습 및 탐지 과정이 아닌 전처리 과정임을 강조한다.

3.1 바이너리 디스어셈블링

명령어를 파악하기 위하여 주어진 악성코드 바이너리와 정상 바이너리를 어셈블리 언어의 형태로 변환한다. IDA pro 디스어셈블링 도구를 이용하여 윈도우 기반 실행 파일을 디스어셈블하여 어셈블리 언어 형태로 변환한다.

3.2 명령어 Opcode 시퀀스 데이터 추출

디스어셈블링을 통해 얻은 바이너리의 피연산자를 포함하는 어셈블리 명령어 시퀀스에서 피연산자 등의 불필요한 정보를 제거하고, Opcode 시퀀스 데이터를 추출한다. 바이너리에 포함된 명령어 순서를 그대로 유지한 채 Opcode 시퀀스 데이터를 추출한다.이때 인텔 기반 프로세서의 명령어 집합의 모든 Opcode (자세히, 인텔 80386)를 지원하도록 한다.

3.3 명령어 Opcode 2-gram 빈도수 추출

명령어 Opcode 시퀀스 데이터로부터 2-gram 빈도수를 추출한다. 여기서 2-gram은 두 개의 연속된 토큰으로 구성된 서열 데이터를 의미한다. 예를 들어, [“mov”, “push”, “push”, “call”, “push”]라는 Opcode 시퀀스 데이터가 존재할 때, 추출된

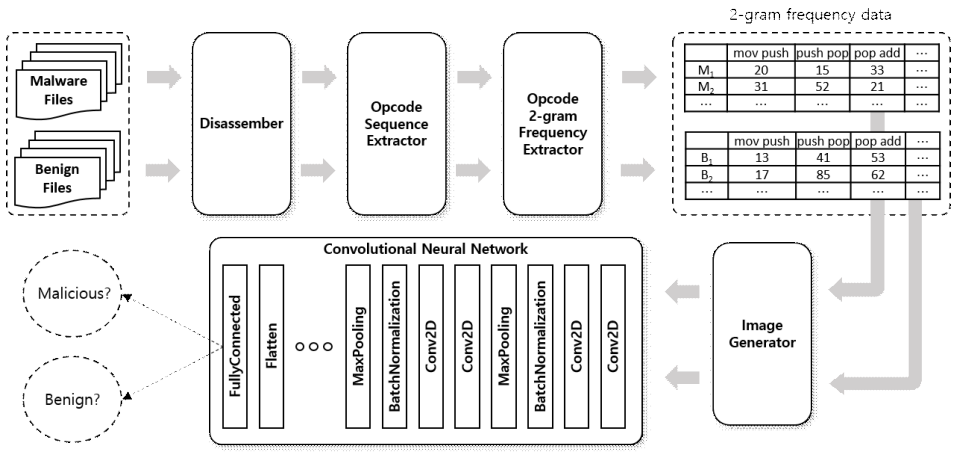


Fig. 1. Overall Processing Flow of Our Proposed Method

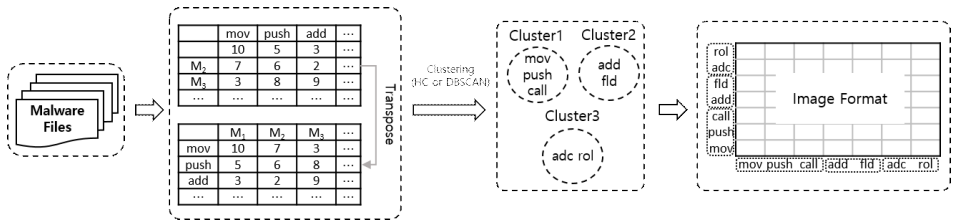


Fig. 2. Image Coordinate Index Mapping via Opcode Frequency-based Clustering

2-gram은 〔〔"mov", "push"〕, 〔"push", "push"〕, 〔"push", "call"〕, 〔"call", "push"〕〕가 된다. 추출을 위하여 크기가 2인 슬라이딩 윈도우를 생성하고, 주어진 서열 데이터를 차례대로 1칸씩 이동하여 2-gram을 추출한다. 추출 시, 2-gram이 Opcode 시퀀스 데이터에서 발견된 빈도수를 함께 기록하도록 한다.

3.4 2-gram 빈도수 기반 이미지 생성

3.4.1 군집화 기반 이미지 좌표 설정 (전처리)

Opcode 2-gram 빈도수 데이터를 이미지화하기 위해 필요한 좌표 인덱스를 설정하는 방법을 설명한다. 3.4.1절의 좌표설정 방법은 실제 학습이나 탐지 과정이 수행되기 이전에 선행되어야 하는 전처리 과정이다. 가로/세로 좌표 인덱스 각각을 어떠한 Opcode와 매핑하느냐에 따라 변환된 이미지의 모습이 달라진다. 그리고 달리 표현된 이미지는 탐지 성능에 영향을 줄 수 있다. 이미지 좌표 설정 과정은 Fig 2.에 나타나있다.

먼저 주어진 악성코드 바이너리들을 디스어셈블하고, 이로부터 Opcode 빈도수 데이터(벡터)를 각 바이너리 별로 추출한다. 추출된 바이너리 별 Opcode 빈도수 벡터를 행으로 하는 행렬을 생성한다. 이후 그 행렬의 전치 행렬을 계산하여 각 행이 Opcode 종류 별 바이너리 내 출현 빈도수를 의미하도록 한다. 각 행은 Opcode를 의미하는 하나의 인스턴스로 설정한다. 이후, 수집한 Opcode 인스턴스들을 이용하여 군집화를 수행하고 유사한 Opcode 인스턴스 그룹을 식별한다. 그리고 동일한 군집에 속한 Opcode를 이미지 좌표에서도 가까이 위치하도록 좌표 인덱스와 매칭한다.

3.4.1.1 군집화 거리함수

Opcode 군집화 과정에서 사용하는 거리함수에 대해 설명한다. 본 연구에서는 피어슨 상관계수를 변형한 거리함수를 이용하여 군집화를 수행한다. 피어슨 상관계수는 수식 1에 나타나 있다. 수식 1의 X와 Y는 데이터(벡터)를 의미하며 $\overline{X}, \overline{Y}$ 는 X, Y의 평균 값을 의미한다.

$$Corr_{xy} = \frac{\sum_{i}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i}^{n}(Y_i - \overline{Y})^2}} \quad (1)$$

$$D_{xy} = 1 - abs(Corr_{xy}) \quad (2)$$

피어슨 상관계수의 범위는 -1부터 1까지이다. 상관 계수는 -1에 가까워질수록 강한 음의 상관관계를 가짐을 뜻하며 1에 가까워질수록 강한 양의 상관관계를 가짐을 뜻한다. 또한 값이 0에 가까워질수록 상관관계가 없다는 것을 의미한다[15]. 최종적인 군집화 거리함수는 수식2에 나타나있다. 피어슨 상관계수의 절댓값을 최대 범위인 1에서 뺌으로써, 상관관계가 높을수록 0에 가까운 결과가 나오도록 하였다.

3.4.1.2 군집화 알고리즘

본 연구에서는 DBSCAN 알고리즘[17]과 계층적 군집화 알고리즘 (Hierarchical Clustering)[16]을 모두 이용하여 군집화를 적용해보고 유용성이 높은 하나의 알고리즘을 택하도록 하였다. 군집화 알고리즘의 유용성은 4절에서 논하며, 두 알고리즘의 적용 방법을 본 절에서 상세 설명한다.

계층적 군집화는 최초 인스턴스 각각을 군집으로 설정하고 차례로 거리가 가까운 두 개의 군집들을 하나로 병합하면서 군집의 개수를 줄여나간다. 군집간의 유사성을 판별할 때는 수식 3의 Group Average Linkage 방식을 사용한다.

$$\triangle(X_i, Y_j) = \frac{1}{|X_i| \ |Y_j|}\sum_{x_i \in X_i}\sum_{y_j \in Y_j} D(X_i, Y_j) \quad (3)$$

수식 3의 방정식은 군집 X와 군집 Y의 모든 조합에 대해 평균값을 구한다. 이때, X_i와 Y_j는 각 군집을 의미하며, $|X_i|$와 $|Y_j|$는 각 군집의 개수를 의미하며, $D(X_i, Y_j)$는 앞서 설명한 거리 함수를 의미한다. DBSCAN은 밀도 기반 공간 클러스터링 알고리즘으로써 임의 인스턴스로부터 사전 정의된 Epsilon(임계 거리) 이내에 최소 개수 이상의 인스턴스들이이 존재할 때 해당 인스턴스를 중심 인스턴스로 정하고 임계 거리 안에 있는 인스턴스를 하나의 군집으로 그룹핑하는 알고리즘이다.

3.4.2 이미지 생성

3.3절에서 설명한 방법으로 추출한 2-gram 빈도수 데이터를 이미지로 변환한다. 3.4.1절에 나타난 방법으로 통해 지정된 X,Y축 좌표 인덱스에 매칭되는 2-gram의 빈도수 값을 각 픽셀에 기록한다. 예를 들어,{{"mov", "push"}: 5, {"push", "push"}: 3, {"push", "call"}:2, {"call", "push"}: 10}이라는 2-gram 데이터가 추출되었을 때, "mov"와 매칭되는 가로축 좌표 인덱스와 "push"와 매칭되는 세로축 좌표 인덱스로 참조되는 위치에는 {"mov", "push"}의 빈도수인 5가 쓰여지게 된다. 위와 같은 과정을 모든 2-gram에 대하여 반복하여 이미지 생성하도록 한다.

3.5 CNN 기반 악성코드 탐지 과정

본 절에서는 제안하는 방법에서 사용하는 CNN에 대해 설명한다. CNN[14]은 이미지 데이터에서 특정 범위의 픽셀 간의 관계성을 포함할 수 있는 딥러닝 알고리즘이다. 제안하는 방법이 사용하는 CNN은 Convolutional Layer, Max Pool Layer, Batch Normalization Layer, Fully Connected Layer로 구성된다. 자세한 CNN 구조 및 파라미터는 4절에서 설명한다.

Convolution Layer는 여러 필터를 이용하여 주어진 입력 데이터를 이동해가며 합성을 수행한다. 합성은 필터를 구성하는 요소에 지정된 가중치와 입력 데이터 요소 값의 선형 결합을 의미한다. Fig 3에 나타난 예에서 알 수 있듯이, 입력의 크기가 5x5이고, 필터의 크기가 2x2로 일 때 필터의 각 요소와 매칭되는 입력 요소가 서로 곱해지고, 곱셈 연산 결과는 모두 더해져 Convolutional Layer의 출력으로 기록된다. 필터가 이동하며 과정을 반복한다.

Max Pool Layer는 필터를 이동시켜가며 매칭되는 입력의 요소 중 가장 값이 큰 값을 선택하고 기록한다. Convolution Layer와 마찬가지로 필터를 이동시켜가며 우세 (Dominant) 한 값을 찾는 과정을 반복한다. Fig 4는 이 과정을 묘사하고 있다.

Batch Normalization[20]은 CNN의 단점을 극복하기 위해 제안되었다. CNN의 층이 깊어짐에 따라서 층이 가지고 있는 평균과 분산 값이 달라져 학습 결과가 안 좋아진다는 단점을 가지고 있다. 따라서 Batch Normalization은 입력 값들의 평균

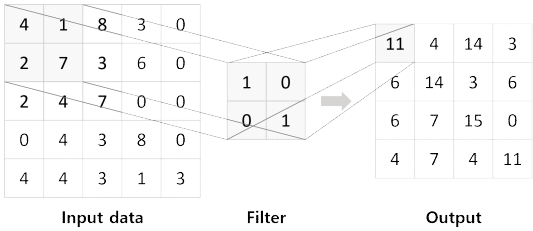


Fig. 3. Example of the Convolution

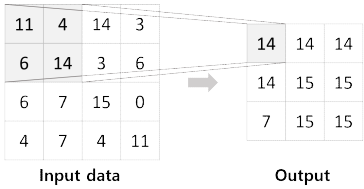


Fig. 4. Example of the Max Pooling

을 0, 분산을 1, 값의 범위를 -1부터 1로 만들어주어 층이 가지고 있는 입력 값을 정규화한다.

Flatten layer는 2차원, 또는 그 이상의 다차원의 이미지를 1차원 배열의 형태로 바꾸므로, 이후에 완전 연결계층을 수행할 수 있는 역할이다.

Fully Connected Layer는 층을 구성하는 모든 뉴런들이 입력 요소와 연결된 형태를 띄는 계층이다. 공간적 정보를 제거하는 역할을 하며, CNN 모델의 최종 결과 값인 분류 결과를 출력하기 전 사전 학습을 수행한다.

Output Layer는 Sigmoid 함수[27]를 활성함수로 사용하는 하나의 뉴런으로 구성된다. Fully Connected Layer의 출력 값을 Output Layer의 입력으로 받아 Output Layer의 가중치 값들과의 선형 결합 결과를 계산하고 해당 값에 Sigmoid 함수를 적용한 결과를 최종 계산한다. 해당 최종 계산 결과 값은 0-1 사이 값으로 CNN의 입력 데이터가 악성코드일 확률 값을 의미한다. 임의 CNN 입력이 0.5 이상의 Sigmoid 계산 결과 값을 가질 때 악성코드라고 판단한다.

IV. 실 험

4.1 데이터 셋

본 논문에서 사용하는 데이터는 한국인터넷진흥원에서 제공한 2018년도, 2019년도 "AI기반 악성코드 탐지" 트랙에 활용된 데이터를 이용한다[18]. 제공하는 PE파일 중, 인텔 80386 PE를 대상으로 하는 코드를 대상으로 했으며, 전체 명령어 개수가 500개 이하인 매우 작은 악성코드, 정상코드를 제외하고 남은 바이너리 중에서 무작위로 악성코드 10,000개, 정상코드 10,000개를 선정하여 사용하였다. 그리고 이미지 크기 즉 사용하는 Opcode 종류의 개수에 따라 탐지 성능이 달라 질 수 있기 때문에 포함 빈도가 높은 상위 10% Opcode 집합, 30% Opcode 집합, 50% Opcode 집합을 먼저 확인하고, 각 Opcode 집합 별로 이미지를 따로 하여 악성코드 탐지 성능을 확인하도록 하였다.

Table 1. CNN train model hyper parameter

Layer	Hyper parameter
Conv2D	Kernel size= (3,3) # of Filters = 32 Activation Function = 'relu'
Conv2D	Kernel size= (3,3) # of Filters = 32 Activation Function = 'relu'
BatchNorm.	-
MaxPool2D	Filter size = (2,2) Strides = (2,2)
Conv2D	Kernel size= (3,3) # of Filters = 64 Activation Function = 'relu'
Conv2D	Kernel size = (3,3) # of Filters = 64 Activation Function = 'relu'
BatchNorm.	-
MaxPool2D	Filter size = (2,2) Strides = (2,2)
Conv2D	Kernel size= (3,3) # of Filters = 128 Activation Function = 'relu'
Conv2D	Kernel size = (3,3) # of Filters = 128 Activation Function = 'relu'
BatchNorm.	-
MaxPool2D	Filter size = (2,2) Strides = (2,2)
Flatten	-
Dense	Units = 512 Activation Function = 'relu'
Dropout	Rate = 0.3
Dense	Units = 1, Activation Function = 'sigmoid'

4.2 CNN 알고리즘 구현 및 파라미터 설정

CNN 알고리즘은 구글의 Tensor flow[19]를 활용하여 구현하였으며 모델 파라미터 및 하이퍼 파라미터는 Table 1에 나타나 있다. 이와 더불어,데이터는 정상코드 이미지 10,000개, 악성코드 이미지 10,000개를 이용했으며, Train, Validation, Test 셋으로 3:1:1 비율로 각각 12,000, 4,000, 4,000개를 할당하여 학습을 진행한다.

4.3 실험 결과

분류 모델에 대한 성능 검증을 위하여 사용한 지표는 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F_1-Score이며, 계산 방식은 수식 4-7에 나타나 있다.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

$$F1-Score = \frac{2*Precision+Recall}{Precision+Recall} \quad (7)$$

각 성능 지표는 TP(True Positive), FN(False Negative), FP(False Positive), TN(True Negative)을 이용하여 계산한다. TP는 실제 악성코드를 악성코드로 탐지한 빈도를 나타내고, FN는 실제 악성코드를 정상으로 판단한 빈도를 나타낸다. FP는 정상프로그램을 악성코드로 예측한 빈도를 의미하고, TN는 정상프로그램을 정상으로 판단한 빈도를 나타낸다.

실험 측정 결과는 Table 2와 Table 3에 나타나 있다. Table 2에는 서로 다른 두 개의 클러스터링 알고리즘(계층적 군집화, DBSCAN)으로 생성한 이미지를 이용한 탐지 결과와 랜덤하게 설정된 좌표에 맞추어 생성된 이미지를 이용한 탐지 결과가 정리되었다. 계층적 군집화를 기반으로 이미지를 생성할 때 다른 방법에 비해 1~2% 정도 높은 정확도를 보이는 것을 확인 할 수 있다.

계층적 군집화의 경우, 하나의 인스턴스를 하나의 군집에 소속되도록 군집화를 시작하고 순차적으로 각 군집과 가장 가까운 군집을 병합하는 방법으로 군집의 크기를 늘려나간다. 이에 따라 최종 결과 도출된 군집 간의 가까운 정도를 정확하게 파악하는 것이 가능하고 동일 군집 내의 인스턴스 간의 유사 정도도 정확히 알 수 있다. 이미지 좌표 상 가까운 위치에 상관도가 높은 Opcode들이 배치되도록 할 수 있다. DBSCAN은 설정된 Epsilon 값 및 클러스터 내 최소 인스턴스 개수가 지정되어, 조건에 부합하지 않는 인스턴스들은 노이즈로 분류된다. 노이즈로 분류된 Opcode들이 비슷한 좌표 위치에 함께 배정된다. 노이즈가 최소로 나오는 Epsilon 값과 최소 인스턴스 개수를 찾아 실험을 하였음에도 비교적 정확도가 낮게 나타났다. 노이즈 Opcode 가 가까운 이미지 좌표 인덱스로 매핑 될 경우, 오히려 랜덤하게 좌표를 설정하는 것보다 더 좋지 않은 성능을 보이는 것을 확인 할 수 있다.

결과적으로 상위 50% 빈도수를 갖는 Opcode들로 계층적 군집화를 수행하여 생성한 이미지(138x138 이미지)를 이용하여 탐지를 하였을 때, 가장 높은 정확도, 0.91가 도출되었다. 이때 Precision, Recall, F_1-score는 각각 0.90, 0.92, 0.91로 나타났으며 Confusion Matrix는 Table 3에서 확인 할 수 있다. Fig 5와 Fig 6은 각각 계층적 군집화 기반 이미지를 CNN으로 학습할 때 측정된 성능 지표를 나타낸다. 학습이 점차 진행됨에 따라 Loss값과 Accuracy 등의 성능 지표가 최적으로 수렴하는 것을 확인 할 수 있다.

Table 2. Malware Detection Accuracy

Type	Image Size		
	46x46	138x138	230x230
Hierarchical Clustering	0.91	0.91	0.89
DBSCAN (Epsilon: 0.3, Min Samples in a Cluster: 5)	0.90	0.89	0.88
Random	0.90	0.90	0.91

Table 3. Confusion Matrix Result

		Predict	
		Positive	Negative
Actual	Positive	1,847 (TP)	153 (FN)
	Negative	210 (FP)	1,790 (TN)

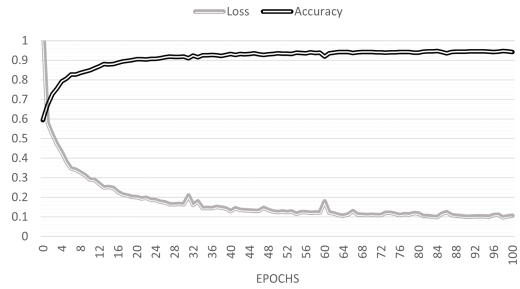


Fig. 5. Accuracy and Loss value Growth along Epochs

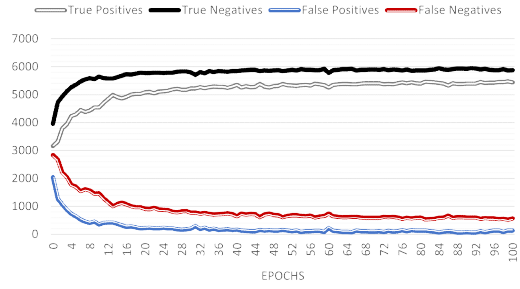


Fig. 6. TP/TN/FP/TN value Growth along Epochs

V. 결 론

본 논문은 CNN을 활용한 악성코드 탐지 방법을 제안한다. 바이너리의 바이트 값을 단순히 이미지화 하지 않고, 명령어 Opcode의 2-gram 빈도수를 이미지화하여 보다 Semantic한 정보를 이미지에 포함할 수 있도록 하였다. 그리고 Opcode 2-gram 빈도수를 이미지로 변환할 때, 이미지의 좌표 인덱스와 Opcode를 매핑시키기 위한 두 가지 군집화 기반의 방법을 동시에 제안하였다. 악성코드 10,000개, 정상프로그램 10,000개를 활용하여 CNN 기반 악성코드 탐지 성능을 확인한 결과 91%의 높은 정확도로 악성코드를 구분해낼 수 있음을 확인할 수 있었다.

References

[1] N. Etaher, G. R. S. Weir, and M. Alazab, "From ZeuS to Zitmo: Trends in Banking Malware," IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 1386-1391, Aug. 2015

[2] Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," IEEE Access, vol. 8, pp. 6249-6271, Jan. 2020

[3] M. Yeo, Y. Koo, Y. Yoon, T. Hwang , J. Ryu, J. Song, and C. Park, "Flow-based malware detection using convolutional neural network," International Conference on Information Networking (ICOIN), pp. 910-913, Jan. 2018

[4] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware Detection with Deep Neural Network Using Process Behavior," IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 577-582, June. 2018

[5] A. Sharma, P. Malacaria, and M. Khouzani, "Malware Detection Using 1-Dimensional Convolutional Neural Networks," IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 247-256, June. 2019

[6] Jaemin Jung, Jongmoo Choi, Seong-je Cho, Sangchul Han, Minkyu Park, and Young-Sup Hwang, "Android malware detection using convolutional neural networks and data section images," Research in Adaptive and Convergent Systems (RACS18), pp. 149-153, Oct. 2018.

[7] Jagsir Singh and Jaswinder Singh, "A survey on machine learning-based malware detection in executable files," Journal of Systems Architecture, vol. 112, Jan. 2021.

[8] Kim HY and Lee DH, "A Study on Machine Learning Based Anti-Analysis Technique Detection Using N-gram Opcode," Journal of the Korea Institute of Information

Security & Cryptology, vol. 32, no. 2, pp. 181-192, Apr. 2022.
[9] YeJin Jeo, Jin-e Kim, and Joonseon Ahn, "Image Generation Method for Malware Detection Based on Machine Learning," Journal of the Korea Institute of Information Security and Cryptology, vol. 32, no. 2, pp. 381-390, Apr. 2022.
[10] Ahmed Bensaoud, Nawaf Abudawaood , and Jugal Kalita, "Classifying Malware Images with Convolutional Neural Network Models," Computer Vision and Pattern Recognition, vol. 22, no. 6, pp. 1022-1031, Nov. 2020.
[11] T.-Y. Wang, and C.-H. Wu, "Detection of packed executables using support vector machines," International Conference on Machine Learning and Cybernetics, vol. 2, pp. 717-722, Jul. 2011.
[12] Z. Salehi, M. Ghiasi, and A. Sami, "A miner for malware detection based on API function calls and their arguments," Artificial Intelligence and Signal Processing (AISP 2012), pp. 563-568, May. 2012
[13] M. Ijaz, M. H. Durad, M. Ismail, "Static and Dynamic Malware Analysis Using Machine Learning," International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp. 687-691, Jan. 2019.
[14] Y. Lecun and L. Bottou and Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 86(11), pp. 2278-2324, Nov. 1998
[15] S. Wright, "Correlation and Causation," Journal of Agricultural Research, vol. 20, pp. 557-585, June 1921.
[16] Frank Nielsen, "Introduction to HPC with MPI for Data Science," Springer, pp. 195-211, 2016.
[17] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," ACM Transactions on Database Systems, vol. 42, no. 3, pp. 1-21, Sept. 2017.
[18] Information Security R&D Introduction, https://www.ksecurity.or.kr/kisis/subIndex/461.do, Accessed at Aug. 2022
[19] M. Abadi, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," Mar. 2015.
[20] S. loffe, and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Proceedings of Machine Learning Research, vol. 37, pp. 448-456, Jul. 2015.
[21] A. Yewale and M. Singh, "Malware detection based on opcode frequency," International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 646-649, May. 2016.
[22] Usukhbayar Baldangombo, Nyamjav Jambaljav and Shi-Jinn Horng, "A Static Malware Detection System Using Data Mining Methods," International Journal of Artificial Intelligence & Application, vol. no. 4, Jul. 2013.
[23] S. Naval, V. Laxmi, M. Rajarajan, M. S. Gaur, and M. Conti, "Employing Program Semantics for Malware Detection," IEEE Transactions on Information Forensics and Security, vol. 10, no. 12, pp. 2591-2604, Dec. 2015.
[24] R. S. Pirscoveanu, S. S. Hansen, T. M. T. Larsen, M. Stevanovic, J. M. Pedersen, and A. Czech, "Analysis of

Malware behavior: Type classification using machine learning," International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), pp. 1-7, Jun. 2015.

[25] Smith Michael, Ingram Joey, Lamb Christopher, Draelos Timothy, Doak Justin, Aimone James, and James Conrad, "Dynamic Analysis of Executables to Detect and Characterize Malware," IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 16-22, Dec. 2018.

[26] K. Aoki, T. Yagi, M. Iwamura, and M. Itoh, "Controlling malware HTTP communications in dynamic analysis system using search engine," Third International Workshop on Cyberspace Safety and Security (CSS), pp. 1-6, Sept. 2011.

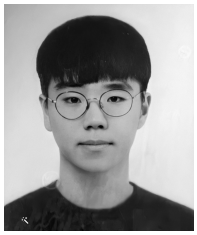
[27] Jun Han and Claudio Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," Springer, vol. 930, pp. 195-201, 1995.

[28] Bakhshinejad, Nazanin, and Ali Hamzeh. "Parallel-CNN network for malware detection," IET Information Security vol. 14, no. 2, pp. 210-219, 2020.

[29] BinZhang, Wentao Xiao, Xi Xiao, Arun Kumar Sangaiah, Weizhe Zhang, and Jiajia Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes," Future Generation Computer Systems, vol. 110, pp. 708-720, 2020.

〈저자소개〉

고 석 민 (Seok Min Ko) 학생회원
2021년 2월: 순천향대학교 정보보호학과 졸업
2021년 3월~현재: 순천향대학교 모빌리티융합보안학과 석사
〈관심분야〉 정보보호, 악성코드 분석, 인공지능



양 재 혁 (JaeHyeok Yang) 학생회원
2019년 2월~현재: 순천향대학교 정보보호학과 학부과정
〈관심분야〉 정보보호, 악성코드 분석, 인공지능

최 원 준 (WonJun Choi) 학생회원
2021년 2월~현재: 순천향대학교 정보보호학과 학부과정
〈관심분야〉 정보보호, 악성코드 분석, 인공지능

김 태 근 (TaeGuen Kim) 종신회원
2011년 2월: 한양대학교 컴퓨터공학과 졸업
2013년 2월: 한양대학교 컴퓨터소프트웨어공학과 석사
2018년 8월: 한양대학교 컴퓨터소프트웨어공학과 박사
2021년 2월: 현대자동차 연구소 책임연구원
2021년 3월~현재: 순천향대학교 정보보호학과 교수
〈관심분야〉 사이버위협인텔리전스, 인공지능 보안, 차량 보안