V-그램: 명령어 기본 블록과 딥러닝 기반의 악성코드 탐지

V-gram: Malware Detection Using Opcode Basic Blocks and Deep Learning

저자
(Authors)

정성민, 김현석, 김영재, 윤명근
Seongmin Jeong, Hyeonseok Kim, Youngjae Kim, Myungkeun Yoon

출처
(Source)

정보과학회논문지 46(7), 2019.7, 599-605(7 pages)
Journal of KIISE 46(7), 2019.7, 599-605(7 pages)

발행처
(Publisher)

한국정보과학회
The Korean Institute of Information Scientists and Engineers

URL

http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE08750801

APA Style

정성민, 김현석, 김영재, 윤명근 (2019). V-그램: 명령어 기본 블록과 딥러닝 기반의 악성코드 탐지. 정보과학회논문지, 46(7), 599-605

ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 46, No. 7, pp. 599-605, 2019. 7
https://doi.org/10.5626/JOK.2019.46.7.599

V-그램: 명령어 기본 블록과 딥러닝 기반의 악성코드 탐지

(V-gram: Malware Detection Using Opcode Basic Blocks and Deep Learning)

정 성 민 † 김 현 석 † 김 영 재 †† 윤 명 근 †††
(Seongmin Jeong) (Hyeonseok Kim) (Youngjae Kim) (Myungkeun Yoon)

요 약 악성코드가 급증하여 기계 학습 기반의 자동 탐지 연구가 중요해지고 있다. 악성코드 실행파일로부터 추출되는 opcode 시퀀스는 악성코드 탐지에 좋은 특징이기 때문에 바이트 기반의 n-그램 처리 기법을 거쳐 기계 학습의 입력 데이터로서 폭넓게 사용되고 있다. 본 논문에서는 처리 속도와 저장 공간 측면에서 기존 n-그램 방식을 크게 향상시키는 기본 블록 단위의 딥러닝 입력 데이터 가공 기법인 V-그램을 새롭게 제안한다. V-그램은 opcode 시퀀스로부터 의미 없는 입력 데이터의 불필요한 생성을 막을 수 있다. 본 논문에서는 64,000개 이상의 실제 정상 및 악성코드 파일을 수집하여 진행한 실험을 통해서, V-그램이 처리 속도와 저장 공간, 그리고 탐지 정확도 측면에서 모두 기존의 n-그램 기법보다 우수하다는 것을 검증하였다.

키워드: 악성코드 탐지, 정적 분석, 디스어셈블, n-그램, 피쳐 해싱

Abstract With the rapid increase in number of malwares, automatic detection based on machine learning becomes more important. Since the opcode sequence extracted from a malicious executable file is useful feature for malware detection, it is widely used as input data for machine learning through byte-based n-gram processing techniques. This study proposed a V-gram, a new data preprocessing technique for deep learning, which improves existing n-gram methods in terms of processing speed and storage space. V-gram can prevent unnecessary generation of meaningless input data from opcode sequences. It was verified that the V-gram is superior to the conventional n-gram method in terms of processing speed, storage space, and detection accuracy, through experiments conducted by collecting more than 64,000 normal and malicious code files.

Keywords: malware detection, static analysis, disassemble, n-gram, feature hashing

· 이 논문은 2018년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2016R1A2B4009083)
· 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No. 2018-0-00429,보안 빅데이터 자동 분석을 위한 실시간 유사도 측정 원천 기술 연구)
· 이 논문은 2018 한국소프트웨어종합학술대회에서 'v-그램: 기본 블록과 딥러닝 기반의 악성코드 탐지'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 국민대학교 컴퓨터공학과
bonopi07@kookmin.ac.kr
kkhs@kookmin.ac.kr
†† 학생회원 : 국민대학교 컴퓨터공학과
corea_kyj@kookmin.ac.kr
††† 종신회원 : 국민대학교 컴퓨터공학과 교수(Kookmin Univ.)
mkyoon@kookmin.ac.kr
(Corresponding author임)

논문접수 : 2019년 3월 4일
(Received 4 March 2019)
논문수정 : 2019년 4월 30일
(Revised 30 April 2019)
심사완료 : 2019년 5월 2일
(Accepted 2 May 2019)

정보과학회논문지 제46권 제7호(2019. 7)

1. 서 론

사이버 공격 기술의 발전과 악성코드 제작자의 적극적인 오픈소스 재사용 등으로 악성코드의 개발 생산성이 높아지고 있으며 발생 건수 또한 급증하고 있다. 더 이상 악성코드 분석가들의 전통적인 수작업 업무에 의존하는 것은 불가능해졌다. 이러한 한계를 극복하기 위해서 기계 학습 기반의 자동 분석 및 탐지 기술이 꾸준히 연구되어 왔으며, 최근에는 딥러닝을 이용한 악성코드 분류 및 탐지 연구가 활발하게 진행되고 있다[1-4].

딥러닝 기반의 악성코드 분석은 네트워크 모델 설계 과정과 학습 입력 데이터 전처리 과정이 성능에 직접적 영향을 미친다[2]. 본 논문에서는 입력 데이터의 전처리 과정을 주요 연구 주제로 다룬다. 입력 데이터를 만드는 방법은 악성코드 파일을 고정된 사이즈의 2차원 이미지로 변환시키거나[3,4], 디스어셈블(disassemble) 과정을 거쳐 opcode 시퀀스(sequence)를 추출하여 특징 벡터로 변환하는 기법이 많이 사용되고 있다[2,5].

최근 기계 학습과 딥러닝 기반의 악성코드 분류 및 탐지 연구는 학습 입력 데이터를 생성하는 기술로서 정적 분석과 n-그램(gram) 기법, 그리고 피쳐 해싱(feature hashing) 기법의 조합을 많이 사용한다[2,6-8]. 이러한 전통적인 학습 데이터 가공 기술을 n-그램 기법이라고 부른다. 본 연구에서는 n-그램 기법이 opcode 시퀀스로부터 의미 없는 학습 데이터를 생산함으로써 처리 속도와 저장 공간, 그리고 분석 정확도 모두에 악영향을 미칠 수 있다는 사실을 밝히고, 이러한 한계를 극복하는 기술로서 V-그램(Variable-length gram)을 새롭게 제안한다.

본 논문에서 제안하는 V-그램 기법은 opcode 시퀀스를 기본 블록(basic block) 단위로 분절하여 워드를 생성하고, 워드 단위로 피쳐 해싱을 적용함으로써 n-그램의 한계를 극복한다. 바이트스트림 기반의 전통적 n-그램 기법과 구분하기 위하여, 기본 블록의 크기가 서로 다르게 생성되는 점에 착안하여 본 논문의 제안 기법을 V-그램으로 명명한다. V-그램의 우수성을 검증하기 위해서 64,000개 이상의 정상 및 악성코드 파일을 수집하여 검증된 악성코드 탐지 딥러닝 모델에 적용시키는 실험을 진행했다. 실험 결과를 통해서 V-그램이 처리 속도와 저장 공간, 그리고 탐지 정확도 측면에서 모두 전통적 n-그램 기반 학습 데이터 전처리 기술보다 우수함을 확인했다.

정적 분석과 피쳐 해싱을 이용하지 않는 악성코드 분석 연구는 본 논문이 다루는 연구 범위에 해당되지 않는다. 즉, 동적 분석이나 이미지 변환 기법 등은 본 논문에서 다루지 않는다.

2장에서는 관련 연구를 소개하고 3장에서는 본 논문에서 제안하는 핵심 아이디어인 V-그램 기법과 개발된 딥러닝 모델을 제안한다. 4장에서는 실험 방법과 결과를 자세히 서술하고 5장에서 결론을 내린다.

2. 관련 연구

딥러닝을 이용한 악성코드 분석 기술은 오랫동안 다양한 방법으로 연구되어왔다. 특히, 악성코드를 딥러닝의 입력 데이터로 사용하기 위해서는 적합한 전처리 과정이 필수적인데, 대표적으로 특징 벡터로 표현하는 방법과 이미지로 표현하는 방법이 많이 사용되고 있다.

2.1 특징 벡터를 활용한 악성코드 분석 연구

J. Saxe 등은 악성코드에서 바이트/엔트로피 히스토그램, PE import, 스트링, PE 메타데이터 정보를 256차원의 벡터로 표현하였는데, 이 특징들은 빠르게 추출할 수 있는 정적 특징이다. 생성된 벡터들은 1,024 차원의 특징 벡터로 표현하여 딥러닝의 입력 데이터로 사용된다. 또한 빠르게 학습 가능한 인공신경망을 이용하여 학습 파라미터가 비교적 적지만 높은 성능 및 정확도를 검증하였다[1].

J. Upchurch 등은 악성코드의 코드 재사용 탐지를 위한 기술을 연구했다. 악성코드는 디스어셈블 과정을 통해 1바이트로 표현한 opcode 시퀀스를 추출하고, n-그램 기법과 피쳐 해싱 기법을 통해 입력 데이터를 생성했다. 이후에는 정상 파일에서 발견되는 특징 데이터를 통해 화이트리스트를 작성하고, 이를 통한 특징 필터링 작업을 수행했다. 최종적으로는 force-based 클러스터링 기법을 적용하여 그룹핑 결과를 보였다[5].

2.2 이미지로 표현한 악성코드 분석 연구

Z. Cui 등은 악성코드를 이미지로 변환하여 코드 변종을 탐지하는 연구를 진행했다. 연구팀은 먼저 악성코드를 바이트 시퀀스로 나타내고, 바이트를 픽셀로 변환하여 2차원 직사각형 이미지를 생성한다. 악성코드의 크기는 모두 다르기 때문에 파일 크기 별로 이미지 생성 기준을 따랐고, 이후에는 고정 크기로 변환하는 작업을 수행했다. 이후에는 이미지 분류 문제와 유사하게 이미지 증강 기술과 무작위 추출 기법, 그리고 CNN (Convolutional Neural Network) 기반 학습 모델을 통해 높은 성능을 검증하였다[3].

S. Ni 등은 딥러닝과 악성코드 이미지를 혼합한 악성코드 탐지 기술인 MCSC(Malware Classification using Simhash and CNN)을 소개했다. MCSC는 악성코드를 이미지로 변환하기 위해 먼저 디스어셈블을 통한 opcode 시퀀스를 추출한다. Opcode 시퀀스는 simhash 인코딩 기법을 거친 후 고정 크기의 이미지로 변환된다. 이후에는 CNN 기반의 딥러닝 학습 모델을 통해 높은 성능을 검증하였다[4].

3. 기본 블록과 딥러닝 기반의 악성코드 탐지 모델

3.1 전반적 구조

딥러닝 기반의 악성코드 탐지 모델은 학습 모드와 테스트 모드로 실행된다. 학습 모드에서 사용되는 모든 파일들은 악성코드와 정상 파일 중 하나로 라벨(label)이 정해져 있으며, 지도 학습 방법을 사용했다. 학습 모델이 완성되면 테스트 파일에 대해서 악성 또는 정상 라벨이 추측되고, 이를 실제 라벨과 비교하여 모델의 정확도를 계산한다. 정확도 비교 실험은 악성코드와 정상 파일을 분류하는 악성코드 탐지 실험과 악성코드의 그룹을 분류하는 악성코드 그룹 분류 실험을 진행한다.

본 논문에서는 실행 파일에 대해서 정적 분석 기술을 적용하여 특징을 추출함으로써 학습 입력 데이터를 생성하는데, 이를 전처리 과정이라고 부른다. 학습용 데이터와 테스트용 데이터는 동일한 방식의 전처리 과정을 거쳐서 고정된 크기의 1차원 벡터로 생성한다. 딥러닝 모델은 이 벡터를 사용해서 그림 1과 같이 학습과 테스트를 진행한다.

3.2 전처리 과정

본 논문에서는 정적 분석을 이용해서 악성코드 실행 파일로부터 opcode 시퀀스를 추출한다. 하나의 opcode는 1바이트를 사용해서 표현하고[5], IDA Pro를 통해 파일의 opcode 시퀀스를 추출하였다[9]. IDA Pro에 의해서 분석이 가능한 정상 및 악성코드 파일은 함수 단위로 구성되고, 각 함수는 다시 기본 블록 단위로 구성된다. 본 논문에서는 각 기본 블록을 opcode의 시퀀스로 표현한다. 그림 2는 임의의 악성코드를 IDA Pro로 분석한 결과 중 일부이다. 분석 대상 악성코드 파일로부터 “함수 A”가 추출되었으며, 이 함수는 세 개의 기본 블록으로 구성되어 있다.

전통적인 n-그램 기법에서는 악성코드로부터 추출된 opcode 시퀀스를 연속된 n개의 바이트로 묶어서 워드를 생산했다. 즉, m 바이트의 시퀀스는 (m-n+1) 개의 워드가 생성한다. 그림 2에서 트라이그램(tri-gram)은 n=3일 때 n-그램 기법으로 생성되는 워드를 집합으로 표현한 것이다.

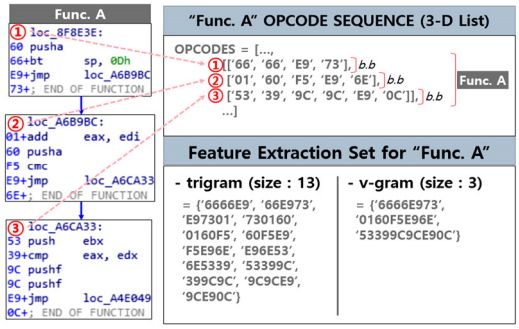

그림 2 IDA Pro 기반 디스어셈블 및 n-그램과 V-그램 비교
Fig. 2 IDA Pro-based disassemble and comparison of n-gram and V-gram

본 논문에서는 이러한 전통적 n-그램 기법이 의미가 없는 워드를 생성시킴으로써 딥러닝의 처리 속도와 저장 공간, 그리고 정확도에 부정적 영향을 미칠 수 있다는 점에 주목하였으며, 이를 개선시킬 수 있는 V-그램 기법을 제안한다. V-그램 기법은 하나의 기본 블록을 하나의 워드로 나타내며 크게 2단계로 진행된다. 먼저, 하나의 기본 블록에 속한 opcode들은 순서대로 이어서 하나의 시퀀스를 만든다. 만들어진 시퀀스는 기본 블록의 길이에 따라 다르므로 가변적이다. 다음으로, 이 시퀀스는 암호학적 해시 알고리즘 중 하나인 SHA-256 알고리즘을 적용하여 하나의 시그니처(signature)를 만든다. 암호학적 해시 알고리즘은 다른 입력에 대해 동일한 출력을 나타내는 확률이 매우 적기 때문에, 시그니처는 기본 블록을 대표하는 유일한 값이 될 것이다. 위 단계들을 모든 기본 블록에 대해 적용한다. 그림 2의 “함수 A”는 기본 블록이 3개이기 때문에, V-그램 기법에 의해 워드는 3개가 생성된다. 13개가 생성되는 트라이그

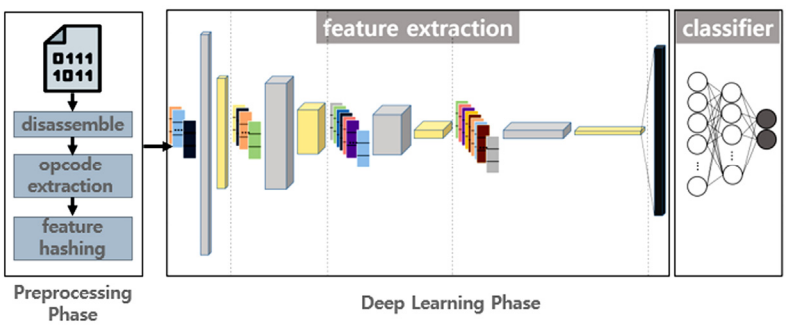

그림 1 딥러닝 모델 구조
Fig. 1 Deep learning model architecture

램과 비교했을 때 비교적 적은 개수이다.

그림 2에서 트라이그램의 'E97301'은 의미를 가지지 않는 불용어로 분류할 수 있다. 왜냐하면, opcode 시퀀스는 기본 블록과 함수 단위로 생성이 되므로, 두 개의 블록에 걸쳐서 생성된 워드는 일반적으로 의미가 없다고 볼 수 있기 때문이다. 이러한 불필요한 원소의 생성은 opcode 시퀀스 기반의 전통적인 n-그램 기법의 한계가 될 수 있다. 우리는 실험을 통해서 V-그램이 n-그램의 한계를 모두 극복할 수 있는 대체 기술이 될 수 있음을 검증한다.

3.3 피쳐 해싱 기반의 특징 벡터 생성 과정

n-그램 또는 V-그램을 통해 생성된 특징 집합은 최종적으로 고정된 크기의 특징 벡터로 변환되어 딥러닝 모델의 입력 데이터로 사용된다. 본 논문에서는 피쳐 해싱 알고리즘을 사용한다. 피쳐 해싱은 고차원의 입력 데이터를 해싱 기법을 적용하여 저차원의 벡터로 사상하는 알고리즘이다. 이 알고리즘은 기계 학습 분야에서 다양한 특징들을 고정 길이의 벡터로 변환시키는데 가장 빠르고 효율적인 방법 중 하나이다[8]. 우선 고정 크기의 벡터를 생성하고 0으로 값을 초기화한다. 입력 특징들에 대해 두 종류의 해시 함수를 적용하는데, 하나는 특징 벡터의 인덱스를 계산하고, 다른 하나는 인덱스가 가리키는 곳의 값을 1 더하거나 빼는 연산을 수행한다. 그림 3은 피쳐 해싱 기반의 특징 벡터를 생성하는 전반적인 과정을 나타낸다. 예를 들어, 특징 집합의 데이터 중 '295786'을 특징 벡터에 적용하고자 할 때, 먼저 해시 함수(hash function) H(m) 을 이용하여 특징 벡터의 인덱스 값을 계산한다. 값을 저장할 인덱스를 구한 이후에는 어떤 값을 넣을지 결정하기 위한 결정 함수(decision function) g(m)를 이용하여 +1, -1 중 하나의 값을 결정한다. 최종적으로 결정된 값을 H(m)에 의해 결정된 벡터의 인덱스에 값을 적용한다. 모든 특징들에 대해 위와 같은 연산을 수행했다면 학습 향상을 위해 범위 [-1, 1] 사이로 값을 정규화한다. 위 과정을 통해 만들어진 특징 벡터는 딥러닝 모델의 입력 데이터로서 활용된다.

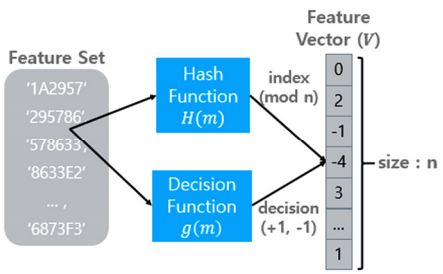

그림 3 특징 벡터 생성 방법

Fig. 3 Feature vector generation

3.4 딥러닝 학습 모델

본 논문에서는 최근 딥러닝의 대표 모델 중 하나인 CNN을 학습 모델로 사용한다. CNN은 합성곱 연산과 풀링(pooling) 연산을 통해 특징을 자동으로 추출하고 분류하는 심층신경망이다. 합성곱 연산은 지역적인 특징을 추출하고, 풀링 연산은 해당 지역의 대푯값을 추출한다. 일반적인 심층 신경망보다 보다 효과적으로 특징을 추출하며, 지역적 정보를 자동 추출하는 특징때문에 근래에 가장 많이 사용되는 학습 모델 중 하나이다[10].

본 논문에서 사용한 딥러닝 모델은 휴리스틱(heuristic)한 실험을 거쳐 악성코드 탐지에 최적화되도록 설계하였다. 학습 파라미터를 가지는 합성곱 필터와 분류기의 은닉층은 처음에 깊게 설정하고, 학습 및 검증 정확도를 통해 과적합(overfitting)을 확인하며 점차 층을 줄여가는 방식을 사용했다. 특히 합성곱 신경망은 점점 필터 수를 늘리는 방식을 채택했으며, 본 연구는 악성코드 데이터 전처리의 성능을 표현하기 위해 필터의 수를 적게 쌓음으로써 적은 학습 파라미터 구조를 유지하였다. 분류기의 은닉층 또한 위와 같은 의도를 포함하여 설계하였다.

본 연구에서 사용하는 CNN은 특징 벡터를 입력 데이터로 사용하기 때문에 1D CNN를 사용했고, 최대 풀링 방식을 적용했다. 활성 함수는 Sigmoid, ReLU, Leaky ReLU를 사용했으나 Leaky ReLU가 성능이 가장 좋았다. 목적 함수는 크로스 엔트로피 함수를 사용했고, 학습 최적화 메커니즘은 adam 옵티마이저를 사용했다. 에폭과 학습률, 배치 크기는 실험에 의한 최적화를 통해 각각 10, 0.0001, 256으로 설정하였다. 학습 파라미터가 있는 층마다 배치 정규화 알고리즘을 적용했기 때문에 드랍아웃은 사용하지 않았다.

표 1은 본 논문에서 사용한 딥러닝 모델의 구조 및 하이퍼파라미터 값을 요약하였다.

표 1 딥러닝 모델 구조 및 하이퍼파라미터

Table 1 Deep learning architecture and hyper parameter

Model		Type	Value
Network architecture	CNN	conv. operation (1D, 1×3 filter)	2,4,8,16
		pooling method	max pooling(1×2)
	ANN	hidden layer	128, 16
other hyper parameters		activation function	Leaky ReLU
		cost function	cross entropy
		epoch	10
		learning rate	0.0001
		optimizer	adam
		batch size	256
		dropout	n.a.

4. 실험 결과

본 논문은 먼저 실험에 사용한 데이터셋과 실험 환경에 대해서 설명한다. 실험은 크게 두 종류를 진행한다. 첫째, n-그램 기법과 V-그램 기법의 성능을 비교하는 실험을 진행한다. 둘째, 딥러닝 모델의 정확도 비교 실험을 진행한다. 세부적으로는 악성코드와 정상 파일을 분류하는 악성코드 탐지 실험과 악성코드의 그룹을 판별하는 그룹 분류 실험을 진행한다.

4.1 데이터 셋과 실험 환경

실험 데이터 셋은 32비트 PE파일의 악성코드 38,261개 와 정상 파일 28,084개로 구성된다. 악성코드는 바이러스사인(VirusSign)에서 수집한 파일 중 바이러스토탈의 리포트 기준으로 카스퍼스키 백신이 악성이라고 판단한 파일로 구성된다[11,12]. 정상 파일은 윈도우즈 운영체제 파일과 어플리케이션 프로그램 파일 중 카스퍼스키 백신이 정상이라고 판단한 파일로 구성된다.

특별히, 악성 파일은 총 6개의 그룹으로 구성되어 있다. 카스퍼스키 백신은 의심 파일에 대해 악성으로 진단할 경우 명확한 진단명이 주어지는데, 진단명의 예시로는 본 실험에서 사용한 임의의 악성코드 진단명 중 하나인 “Trojan-Downloader.Win32.Hmir.dnm”이다. 예시와 같이, 카스퍼스키 백신은 점(‘.’)을 기준으로 일반적으로 최대 4개의 정보를 포함하는 진단명을 적용하는 것을 확인하였다. 첫 번째 정보는 악성코드의 행위 정보를 나타내며, 예를 들면 “Worm”, “Trojan-Ransom”, “Not-a-virus:Adware”, “Rootkit” 등이 있다. 두 번째 정보는 악성코드의 동작 운영체제 정보이며, “Win32”, “Win64” 등이 있다. 세 번째 정보는 악성코드의 상세 명칭 혹은 불리우는 명칭인데, “WannaCry”, “Hmir” 등 다양하다. 마지막 정보는 악성코드의 변종 분류 정보이다. 본 실험에 사용한 악성코드들은 진단명의 첫 번째 정보만을 활용하였고, 활용한 진단명 그룹은 (“not-a-virus:Downloader”, “Backdoor”, “Virus”, “Trojan-Ransom”, “Worm”, “Trojan”)이다. 본 실험에서 사용한 악성코드의 그룹 당 개수 분포는 그림 4와 같다. Virus 그룹이 가장 많고 Backdoor 그룹의 수가 가장 적다.

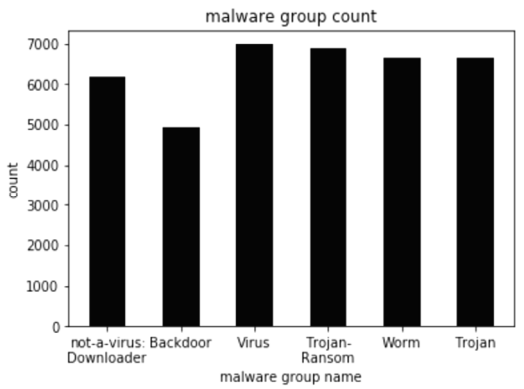

그림 4 악성코드 그룹 분포
Fig. 4 Distribution of malware groups

실험 환경은 Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz, DDR4 128GB RAM 사양의 PC를 사용했다. 딥러닝을 위한 GPGPU는 VGA NVIDIA Titan Xp를 사용했다. 모든 스크립트는 python 3.6으로 구현하였고, IDA Pro 7.0 버전과 텐서플로우(Tensorflow) 1.8 버전을 사용하였다[9,13].

4.2 데이터 전처리 속도 비교

표 2는 특징 추출 기법에 따른 2,048차원의 특징 벡터를 생성하는데 소요되는 시간을 비교한다. n-그램은 n=3, 4, 5일 때를 비교하였고, V-그램은 2장에서 기술한 방식대로 생성하였다. V-그램 기법은 악성코드 전처리 시간을 최대 약 26% 단축했고, 정상 파일은 약 59% 단축했다. 악성코드와 정상 파일의 전처리 시간 차이가 발생하는 이유는, 라벨 별 opcode 입력 데이터의 평균 크기 차이가 있기 때문이다. 실제로 악성코드 데이터의 평균 크기는 약 164KB이고, 정상 파일 데이터의 평균 크기는 1,144KB로 입력 데이터 수 차이에 의해 전처리 시간이 영향을 받은 것을 확인하였다.

표 2 전처리 시간 비교(n-그램, V-그램)
Table 2 Comparison of preprocessing time (n-gram vs V-gram)

	preprocessing method	time(sec)	number of features
malware	3-gram	1,939	619,087,009
	4-gram	1,918	619,052,463
	5-gram	1,898	619,017,998
	V-gram	1,403	102,336,292
benign ware	3-gram	8,637	3,123,367,108
	4-gram	8,624	3,123,339,118
	5-gram	8,612	3,123,311,385
	V-gram	4,070	548,421,684

실험 결과를 통해 V-그램 기법의 특징 수가 n-그램 기법보다 훨씬 적은 것을 확인했으며, 저장 공간 및 전처리 성능이 향상됨을 확인하였다.

4.3 딥러닝 탐지 모델 정확도 비교 실험

본 논문에서는 추출 기법과 특징 벡터 크기에 따라 다른 입력 데이터를 통해 악성코드 탐지 정확도를 계산하였다. 또한, k=5인 k-fold 교차 검증 기법을 적용하여 실험 데이터 셋에 대한 딥러닝 성능을 검증하였다. 설계한 딥러닝 모델의 네트워크가 복잡하지 않기 때문에, 학

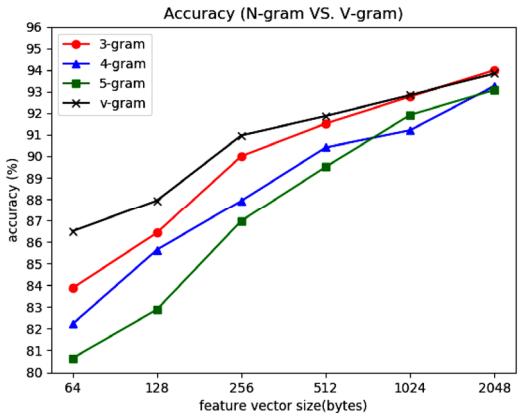

그림 5 딥러닝 탐지 모델 정확도 비교

Fig. 5 Comparison of model detection accuracy

습 및 검증 시간도 빠를 뿐 아니라 입력 데이터가 정확도에 어떻게 영향을 미치는지 파악할 수 있다.

그림 5는 추출 기법에 따른 실험 정확도를 나타낸다. 특징 벡터의 크기가 클수록 높은 정확도를 나타내며, V-그램이 n-그램보다 높은 분류 정확도를 보여주고 있다. 또한 n-그램 실험 중에서는 3-그램일 때 가장 정확도가 높다. n-그램 기법을 통한 특징 벡터는 크기가 작아질수록 정확도가 큰 폭으로 나빠지고 있지만 V-그램 기반의 특징 벡터는 성능 저하가 그보다는 완만하게 진행됨을 보여준다.

학습 모델의 과적합을 평가하는 방법 중 하나는 모델의 학습 정확도와 검증 정확도를 비교했을 때, 검증 정확도가 지나치게 낮아지는지를 확인하는 것이다. 그림 6은 V-그램으로 생성한 2,048 차원의 특징 벡터를 학습 모델의 입력 데이터로 사용했을 때, 학습과 검증에 대한 에폭 별 정확도를 비교하는 그래프이다. 에폭에 따른 학습과 검증 정확도가 올라가고 있으며, 특히 검증 정확도

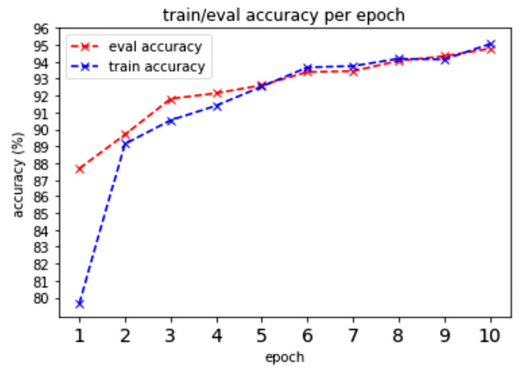

그림 6 에폭에 따른 학습 모델 정확도 비교

Fig. 6 Comparison of training/evaluation accuracy per epoch

가 일관적으로 증가하고 있는 것을 확인하였고, 설계한 학습 모델이 과적합되지 않은 신뢰도가 높은 모델임을 검증하였다.

4.4 딥러닝 분류 모델 정확도 비교 실험

그림 7은 위 실험에서 정확도가 높았던 3-그램과 V-그램에 대해 특징 벡터의 크기에 따른 악성코드 그룹 분류(총 6개 그룹) 실험 정확도를 나타낸다. 앞선 실험 결과와 비슷하게 특징 벡터의 크기가 커질수록 V-그램이 n-그램보다 높은 분류 정확도를 보여주고 있다. 학습 모델이 특정 그룹을 정확히 맞출수록 분류 정확도는 올라가기 때문에, 탐지 뿐만 아니라 분류 성능 또한 V-그램 기법이 n-그램 기법보다 좋다는 것을 검증하였다. 또한 V-그램 기법은 특징 벡터의 크기가 커질수록 n-그램 기법보다 넓은 폭으로 정확도가 올라가는 것을 확인하였다.

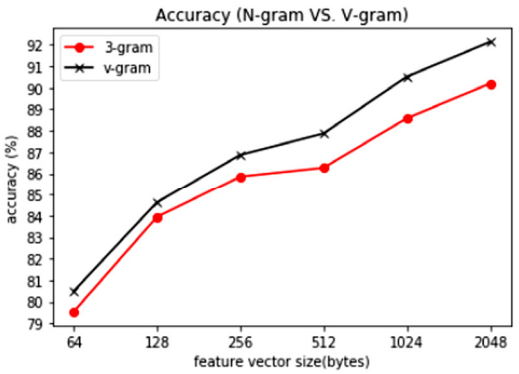

그림 7 딥러닝 분류 모델 정확도 비교

Fig. 7 Comparison of model classification accuracy

두 실험을 통해 특징 벡터의 크기가 적당히 클수록 좋은 학습 정보가 생성됨을 알 수 있고, V-그램 추출 기법은 특징 벡터에 양질의 정보를 반영시킨다는 것을 검증하였다. 결론적으로 표 2와 그림 5, 그림 7로부터 본 논문에서 제안하는 V-그램 방식이 전통적인 n-그램 방식보다 학습 데이터의 처리 속도, 저장 공간, 그리고 딥러닝 모델의 정확도 향상 측면에서 모두 더 우수함을 확인하였다.

5. 결 론

본 논문에서는 가변적 특징 추출 알고리즘인 V-그램 기법을 제안하였고 특징 벡터를 입력으로 하는 딥러닝 기반 악성코드 탐지 모델에 V-그램을 적용하였을 때 더 높은 성능 및 정확도를 보인다는 것을 실험을 통해서 확인하였다. V-그램 추출 기법은 악성코드 분석 뿐 아니라 opcode 시퀀스를 사용하는 다른 분야에서도 적

용이 가능하다. 또한, 딥러닝 모델을 최적화한다면 더 높은 악성코드 탐지 정확도가 나올 것으로 기대된다.

References

[1] J. Saxe, and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 11-20, Oct. 2015.

[2] S. Jeong, S. Lee, J. Jeong, and M. Yoon, "Deep Learning and Malware Analysis," *Communications of the Korean Institute of Information Scientists and Engineers*, Vol. 36, No. 2, pp. 37-42, Feb. 2018.

[3] Z. Cui, F. Xue, Y. Cao, and G. Wang, "Detection of Malicious Code Variants Based on Deep Learning," *IEEE Trans on Industrial Informatics*, Vol. 14, No. 7, pp. 3187-3196, Jul. 2018.

[4] S. Ni, Q. Qian, and R. Zhang, "'Malware identification using visualization images and deep learning," *Elsevier Computers and Security*, Vol. 77, pp. 871-885, Aug. 2018.

[5] J. Upchurch and X. Zhou, "First Byte: Force-Based Clustering of Filtered Block N-grams to Detect Code Reuse in Malicious Software," *2013 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE)*, pp. 68-76, Oct. 2013.

[6] N-gram [Online]. Available: https://en.wikipedia.org/wiki/N-gram.

[7] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based detection of new malicious code," *COMPSAC 04: Proceedings of the 28th Annual International Computer Software and Applications Conference Workshops and Fast Abstracts*, pp. 41-42 Oct. 2004.

[8] K. Weinberger and J. Attenberg, "Feature Hashing for Large Scale Multitask Learning," *ICML*, pp. 1113-1120. Feb. 2009.

[9] IDA Pro [Online]. Available: https://www.hex-rays.com/products/ida/.

[10] I. GoodFellow, Y. Bengio, and A. Courville, "Deep Learning," The MIT Press, 2016 [Online]. Available: http://www.deeplearningbook.org.

[11] Virussign [Online]. Avaliable: http://www.virussign.com.

[12] Virustotal [Online]. Available: http://www.virustotal.com.

[13] Tensorflow [Online]. Available: https://www.tensorflow.org.

정 성 민

2018년 2월 국민대학교 컴퓨터공학부 학사
2018년 3월~현재 국민대학교 컴퓨터공학과 석사과정. 관심분야는 기계학습, 데이터마이닝, 빅데이터

김 현 석

2014년 3월~현재 국민대학교 컴퓨터공학부 학사과정. 관심분야는 정보보호, 기계학습, 빅데이터

김 영 재

2019년 2월 국민대학교 소프트웨어학부 학사. 2019년 3월~현재 국민대학교 컴퓨터공학과 석사과정. 관심분야는 인공지능, 데이터마이닝, 빅데이터

윤 명 근

1996년 2월 연세대학교 컴퓨터과학과 학사
1998년 2월 연세대학교 컴퓨터공학과 석사
2008년 12월 University of Florida, 컴퓨터공학 박사. 1998년 1월~2010년 2월 금융결제원 과장. 2010년 3월~현재 국민대학교 컴퓨터공학과 부교수. 관심분야는 컴퓨터&네트워크 보안, Randomized Algorithm, 지능형 보안, 금융보안