Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.4, Aug. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.4.847

메모리 추가 신경망을 이용한 희소 악성코드 분류

강 민 철,† 김 휘 강‡
고려대학교 정보보호대학원

Rare Malware Classification Using Memory Augmented Neural Networks

Min Chul Kang,† Huy Kang Kim‡
Graduate School of Information Security, Korea University

요 약

악성코드의 수가 가파르게 증가하면서 기업 및 공공기관, 금융기관, 병·의원 등을 타깃으로 한 사이버 공격 피해 사례가 늘어나고 있다. 이러한 흐름에 따라 학계와 보안 업계에서는 악성코드 탐지를 위한 다양한 연구를 진행하고 있다. 최근 들어서는 딥러닝을 비롯해 머신러닝 기법을 적용하는 형태의 연구가 많이 진행되는 추세다. 이 중 합성곱 신경망(CNN: Convolutional Neural Network), ResNet 등을 이용한 악성코드 분류 연구의 경우에는 기존의 분류 방법에 비해 정확도가 크게 향상된 것을 확인할 수 있다. 그러나 타깃 공격의 특징 중 하나는 사용된 악성코드가 불특정 다수를 상대로 광범위하게 퍼뜨리는 형태가 아닌, 특정 대상을 타깃으로 한 맞춤형 악성코드라는 점이다. 이러한 유형의 악성코드는 그 수가 많지 않기 때문에 기존에 연구되어온 머신러닝이나 딥러닝 기법을 적용하기에 한계가 있다. 본 논문은 타깃형 악성코드와 같이 샘플의 양이 부족한 상황에서 악성코드를 분류하는 방법에 대해 다루고 있다. 메모리가 추가된 신경망(MANN: Memory Augmented Neural Networks) 모델을 이용하였고 각 그룹별 20개의 소량 데이터로 구성되어 있는 악성코드 데이터셋에 대해 최대 97%까지 정확도로 분류할 수 있음을 확인하였다.

ABSTRACT

As the number of malicious code increases steeply, cyber attack victims targeting corporations, public institutions, financial institutions, hospitals are also increasing. Accordingly, academia and security industry are conducting various researches on malicious code detection. In recent years, there have been a lot of researches using machine learning techniques including deep learning. In the case of research using Convolutional Neural Network, ResNet, etc. for classification of malicious code, it can be confirmed that the performance improvement is higher than the existing classification method. However, one of the characteristics of the target attack is that it is custom malicious code that makes it operate only for a specific company, so it is not a form spreading widely to a large number of users. Since there are not many malicious codes of this kind, it is difficult to apply the previously studied machine learning or deep learning techniques. In this paper, we propose a method to classify malicious codes when the amount of samples is insufficient such as targeting type malicious code. As a result of the study, we confirmed that the accuracy of 97% can be achieved even with a small amount of data by applying the Memory Augmented Neural Networks model.

Keywords: Malware Classification, Visualization, Memory Augmented Neural Network

Received(04. 02. 2018), Modified(1st: 07. 05. 2018, 2nd: 07 .23. 2018), Accepted(07. 23. 2018)

† 주저자, pukekaka@korea.ac.kr
‡ 교신저자, cenda@korea.ac.kr(Corresponding author)

I. 서 론

스마트폰과 같은 IT 기술이 발전함에 따라 금융, 소통, 쇼핑 등 실생활에 밀접하게 닿아있는 일들을 인터넷으로 처리하는 시대가 도래했다. 특히 기업들이 대부분의 정보를 디지털 형식으로 저장하고 처리함에 따라 금전적인 이득을 취하기 위한 기술도 고도화되고 정교해졌다. 최근에는 암호화폐 채굴, 사물인터넷 등 최신 트렌드에 맞춰 랜섬웨어, 비트코인 마이너와 같은 악성코드가 유포되고 있다. 악성코드는 개인뿐만 아니라 회사와 공공기관에게도 심각한 위협이 된다. 2017년 2분기 한국인터넷진흥원 사이버 위협 동향 보고서[1]에 따르면 지난 2010년부터 국내 방위산업체를 대상으로 한 공격이 꾸준히 발생하고 있다. 방위산업체의 경우, 국가 안보와 밀접하게 연관된 정보를 다루는 기관이기 때문에 경쟁 관계 혹은 적대 관계에 있는 국가에 의한 행위일 가능성도 배제하기 어렵다. 해당 보고서에 따르면 일부 공격 그룹은 방위산업체뿐 아니라 정치 및 외교 관련 기관에도 공격을 가하고 있는 것으로 나타났다. 이는 산업 기밀 탈취 수준을 넘어 국가 안보를 위협하는 행위로써 지속적으로 활동 영역을 넓혀 가고 있는 것으로 추정된다.

이와 같은 악성코드들은 안티바이러스 제품의 탐지를 우회하기 위해 일부 코드들을 조금씩 바꾸기 때문에 기존 시그니처 방식으로 대응하기 어려운 점이 있다. 이러한 한계점을 극복하기 위해 학계와 보안업계에서는 데이터 마이닝, 머신러닝 등 다양한 방법을 활용한 연구를 진행하고 있으며, 최근에는 합성곱 신경망(CNN: Convolutional Neural Network), 순환 신경망 (RNN: Recurrent Neural Network), ResNet(Deep Residual Networks) 과 같은 딥러닝을 이용한 분류 방법에 대한 연구도 활발히 이뤄지고 있다. 그러나 기업 및 공공기관, 금융기관, 병·의원 등을 타깃으로 한 사이버 공격은 일반적인 사이버 공격과 다른 패턴을 보이는 것이 변수다. 여러 요인 가운데 핵심 변수는 불특정 다수에게 유포되는 악성코드와 다르게 그 수가 많지 않다는 것이다. 이러한 특징은 최근 많이 활용되었던 머신 러닝과 딥러닝 알고리즘을 적용하기 어렵게 만든다.

본 논문에서는 샘플의 수가 많지 않은 악성코드를 분류하기 위해 이미지화 하고 메모리가 증가된 신경망(MANN: Memory Augmented Neural Networks) 모델을 적용한 분류 방법을 제안한다.

II. 관련 연구

악성코드는 2007년을 기점으로 증가하기 시작하여 2010년경부터 급증한 것으로 나타났다[2]. 또한 국내 Anti-Virus 업체의 통계에 따르면 국내에서만 하루에 수십만 개의 새로운 악성코드들이 수집되고 있다[3]. 이렇게 매년 악성코드가 급증하는 데 반해, 보안 기술 발전 속도는 상대적으로 이를 따라잡지 못하고 있는 실정이다. Anti-Virus 업체 역시 기존의 대응 방식으로 수많은 악성코드를 처리하기에 한계를 느껴 악성코드 자동화 처리를 위한 시도를 지속해왔다. 특히 악성코드를 이미지화하고 분류하는 방식은 그 효율성 및 성능 측면에서 많은 장점이 있어 꾸준히 연구되고 있다.

Nataraj 외[4]는 악성코드를 분류하기 위해 악성코드 자체를 시각화하는 방법을 최초로 제안하였다. Gabor filter를 이용하여 악성코드를 이미지화한 후, K-NN(K-Nearest Neighbors) 알고리즘을 이용하여 25개의 다른 패밀리에 속해있는 9,458 악성코드를 97.18%의 정확도로 분류하였다.

Seon Hee Seok 외[5]는 악성코드를 이미지화한 후, CNN 신경망 알고리즘을 이용해 분류하였다. 총 20,000개의 샘플을 9개의 다른 패밀리로 분류하는데, 98%의 정확도를 나타냈다. D. Gibert 외[6], X. Meng 외[7] 연구에서도 CNN을 활용한 분류 방법으로 높은 정확도를 얻는데 성공했다.

Ajay Singh 외[8] 또한 악성코드 바이너리를 32 * 32 이미지로 변환 후, CNN과 ResNet 을 이용한 분류방법을 연구했다. 이 분류 방법으로 20개의 다른 클래스에 속한 30,106 개의 악성코드를 분류했으며, CNN은 95.24%, ResNet은 98.21%의 정확도를 얻었다.

Kyoung Soo Han 외[9]는 악성코드의 정적 분석과 동적 분석을 통해 추출한 opcode 시퀀스를 이용해 이미지 매트릭스를 생성하고 이를 시각화하여 분석하는 방법을 제안하였다. 추가로 악성코드 분석가가 악성코드를 시각적으로 분석하고 활용할 수 있도록 분석 도구를 구현하였다.

Microsoft Malware Classification Challenge in 2015 에서 9개의 다른 분류에 속하는 500기가 분량의 악성코드 데이터셋을 제공했다.

이 대회에서 1등을 한 Xiaozhou [10]는

Opcode N-Grams (N=2,3,4), Segment Line Count, asm 파일 이미지화를 이용하여 99%의 정확도로 분류하는데 성공했다.

Lim 외[11]은 악성코드의 시각화된 이미지 입력값과 Word2Vec 알고리즘을 이용한 API BoW(Bag of Words) 입력 값을 혼합하여 학습하는 모델을 제안하였다. 총 20,000개의 샘플에 대해 시각화된 이미지는 합성곱 신경망을 이용하여 학습하고, 각 샘플들의 API는 Word2Vec과 다층 퍼셉트론(Multi-layer Perceptron) 을 이용하여 학습하였다. 이 혼합 모델은 악성코드와 정상샘플을 분류하는데 95.72%의 정확도의 결과를 보여주었다.

T. H.-D. Huang 외[12]는 이미지화 하는 방법을 Android 악성코드에 적용하였고 변환된 컬러 이미지와 합성곱 신경망 알고리즘을 이용해 악성코드를 분류하였다.

III. 배경 지식

3 장에서는 메모리를 사용하는 신경망 모델의 구조와 관련된 지식들을 소개한다.

3.1 Neural Turing Machines (NTM)

Alex Graves 외[13]가 제안한 NTM(Neural Turing Machines)은 크게 신경망으로 이루어진 컨트롤러(controller)와 가중치 벡터(weight vector)를 저장하는 메모리로 구성되어 있다. NTM의 전체 아키텍처는 Fig. 1.과 같으며, 메모리와 컨트롤러 사이에는 읽기 헤드(read heads)와 쓰기 헤드(write heads)로 구성되어 있다.

기본적으로 컨트롤러는 Feed-Forward 신경망

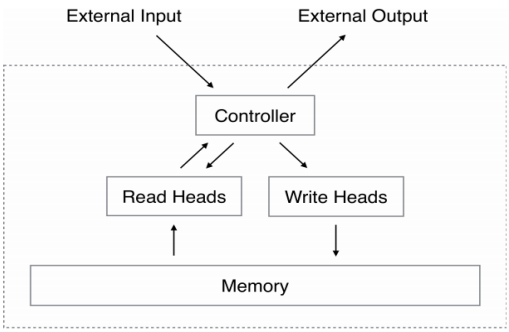

Fig. 1. Neural Turing Machine Architecture

또는 LSTM (Long-Short Term Memory) 신경망을 사용한다. 이렇게 구성된 컨트롤러의 입력 데이터는 외부 입력값(external input)과 기존의 메모리에 존재하던 값이 합쳐진 상태로 이루어진다. 이때 메모리에 저장된 값은 읽기 헤드를 이용해 읽는다. 이어서 컨트롤러는 메모리에 저장될 벡터와 메모리에서 읽어야할 벡터를 위해 필요한 값들을 출력한다. 이렇게 출력된 값들은 읽기 헤드와 쓰기 헤드를 구성하는데 이용된다.

읽기/쓰기 헤드에서 값을 읽고 저장하기 위해서는 메모리의 위치를 선택해야 한다. 위치를 선택하기 위한 매커니즘은 크게 내용 기반 주소 지정 방식(content-based addressing) 과 위치 기반 주소 지정 방식(location-based addressing) 으로 나뉘진다. 위치 주소 지정 방식은 내부적으로 Interpolation, Convolutional Shift, Sharpening 과정으로 이루어져 있다.

컨트롤러가 메모리에서 위치를 찾고 연산하는 전체의 과정은 미분이 가능하기 때문에 기울기 하강 방법을 통해 학습이 가능하다. 이렇게 학습된 NTM 모델은 임의의 시퀀스 데이터를 입력받아 입력 값을 그대로 출력하는 Copy Problem 실험과 이를 더 확장하여 반복하는 Repeat Copy 실험에서 LSTM 보다 더 좋은 성능을 보여준다. 더 나아가 정렬 알고리즘과 같은 간단한 알고리즘을 학습하는 것도 가능하다.

3.2 Memory Augmented Neural Networks (MANN)

A. Santoro 외[14]가 제안한 MANN(Memory Augmented Neural Networks) 모델은 기존의 NTM 모델에서 일부를 변형한 모델이다. NTM 모델과의 가장 큰 차이점은 내용 기반의 주소 지정 방식만을 사용하고 LRUA (Least Recent Used Access) 모듈이 추가된 점이다.

NTM과 같이 메모리를 추가한 모델은 모든 경우의 수를 학습시키는 것이 아니다. 사전에 알려주지 않은 정보라도 과거에 저장된 메모리의 값을 이용해 활용 및 추론할 수 있다는 장점이 있다. 이때 메모리 업데이트의 시기를 빠르게, 혹은 느리게 조절하면 단기적인 기억은 물론, 장기적인 기억도 추출이 가능하도록 변경할 수 있다. 이러한 점을 이용하면 데이터의 유형을 분류하는 방법을 배우는 일종의 메타 학습

뿐만 아니라, 정보들의 연관성에 대한 단기 기억에 집중하여 데이터가 소량이거나 충분하지 않을 때에도 효과적으로 활용할 수 있다. MANN 모델의 전체적인 아키텍처는 Fig.2 와 같다.

MANN 모델의 읽기 메커니즘은 메모리에서 관련성 있는 정보를 판단하기 위해 먼저 컨트롤러의 출력 값인 키 벡터(k_t)와 메모리의 각 행($M_t(i)$)을 이용해 코사인 유사도(1)를 구한다. 여기서 i는 메모리 번지를 의미한다.

$$K(k_t, M_t(i)) = \frac{k_t \bullet M_t(i)}{\| k_t \| \bullet \| M_t(i) \|} \quad (1)$$

코사인 유사도 값(K)을 입력받는 소프트 맥스 함수(2)를 이용해 가장 관련 있는 읽기 가중치 벡터(w_t^r)를 구한다.

$$w_t^r = \frac{\exp(K(k, M(i)))}{\sum_j \exp(K(k, M(j)))} \quad (2)$$

읽기 가중치 벡터(w_t^r)와 메모리의 각 행을 이용해 최종 읽기 벡터(r_t)를 계산하고(3) 이 값은 다시 다음 입력 값으로 이용된다.

$$r_t \leftarrow \sum_i^R w_t^r(i) M_t(i) \quad (3)$$

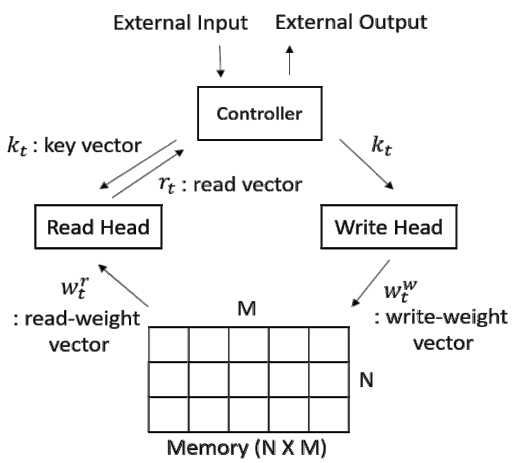

Fig. 2. MANN Architecture

쓰기 메커니즘인 라벨과 입력 값의 쌍을 메모리에 저장하기 위한 과정은 NTM의 Interpolation 과정과 같다. 차이점은 쓰기 가중치(w_t^w) 값을 시그모이드 함수(σ)를 이용하여 이전 단계의 쓰기 가중치(w_{t-1}^w)와 최근 사용 가중치 (w_{t-1}^{lu}) 와의 값으로 결정하는 것이다. 여기서 α값은 상수 값이다(4).

$$w_t^w \leftarrow \sigma(\alpha) w_{t-1}^w + (1 - \sigma(\alpha)) w_{t-1}^{lu} \quad (4)$$

컨트롤러의 출력 값인 키 벡터(k_t)와 메모리의 각 행을 이용하여 저장 위치를 구한다(5).

$$M_t(i) \leftarrow M_{t-1}(i) + w_t^w(i) k_t \quad (5)$$

감마(λ)는 상쇄 상수 값으로 이전 단계에서 사용된 가중치(w_{t-1}^u) 값에 영향을 준다. 추가로 읽기 가중치 벡터(w_t^r)와 쓰기 가중치 벡터(w_t^w)와 함께 사용했던 가중치 값(w_t^u)을 결정한다(6).

$$w_t^u \leftarrow \gamma w_{t-1}^u + w_t^r + w_t^w \quad (6)$$

사용했던 가중치 값(w_t^u)에서 n번째로 작은 값(m) 에 따라 최근 사용했던 가중치 값(w_t^{lu})을 결정한다(7).

$$w_t^{lu}(i) = \begin{cases} 0 \text{ if } w_t^u(i) > m(w_t^u, n) \\ 1 \text{ if } w_t^u(i) \leq m(w_t^u, n) \end{cases} \quad (7)$$

결과적으로 MANN 모델은 NTM 모델이 사용하였던 위치기반의 주소지정 방식을 사용하지 않은 대신, LRUA 모듈을 추가하여 정보와의 연결성에 집중되도록 변형하였다. 결국, LRUA 모듈은 미사용된 메모리 위치에 새로운 정보를 쓰고, 마지막으로 사용된 위치에 최근 이용된 정보 또는 연관 정보를 다시 저장하게 하는 모듈로서, 정보 연결성을 강조하는 기능을 한다. 이렇게 학습된 신경망은 정보들의 연관성을 이용한 분류 방법을 학습하기 때문에 Omniglot 데이터셋[15]과 같은 소량의 데이터를 분류할 때 좋은 성능을 보여준다.

IV. 제안하는 방법

4.1 악성코드 이미지화

악성코드 패밀리를 분류하기 위해서 각 패밀리별 악성코드의 특징을 표현하는 작업이 선행된다. 본 논문에서는 Fig.3 의 순서대로 악성코드 바이너리에 대한 이미지 변환 작업을 먼저 수행한다.

타깃형 악성코드들은 소량이라는 특성 외에도, 코드의 진행 방식이나 흐름에서 유사성이 있다. 이러한 특성이 나타나는 이유는 제작자들이 다수가 아니라는 점과 주로 특정 그룹에 의해 공격이 이루어지기 때문으로 판단된다. 한국인터넷진흥원 사이버 위협 동향 보고서[1], 시만텍[16], 인포섹[17] 리포트를 보면 많은 부분에서 동일한 코드 흐름이 발견되는 것을 확인할 수 있다.

본 논문에서는 악성코드 바이너리에서 코드 영역을 추출하여 생성한 이미지를 특징으로 사용한다. 실험 시, 그레이 스케일로 변환된 이미지를 그대로 사용하면 많은 메모리와 연산량이 필요하기 때문에 효율성과 성능을 높이기 위하여 그레이 스케일 이미지의 크기를 20 * 20 으로 다운사이징 하였다.

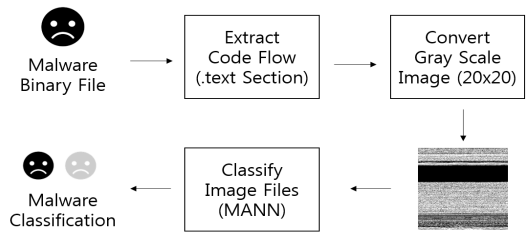

Fig. 3. Data Processing

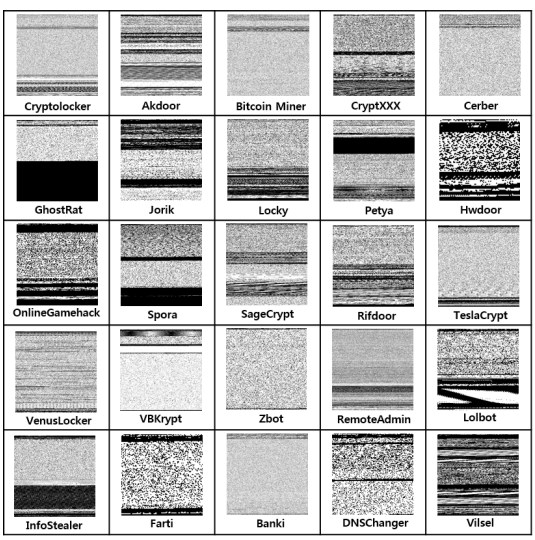

Fig. 4. Malware Image in Mal60 Dataset

Fig.4는 Mal60 데이터셋의 60개의 패밀리 중 25개 패밀리에 속하는 악성코드들을 이미지 변환하였다. 결과적으로, 서로 다른 그룹에 속한 악성코드들은 이미지 패턴이 동일하지 않고 고유한 패턴을 가지고 있음을 확인할 수 있다.

4.2 분류 모델 설정

CNN 모델로 좋은 성능을 보이는 연구들의 경우 최소 수백 개의 샘플들로 학습을 하였다. 또한 Seon Hee Seok 외[5] 연구에서 샘플들이 100개 미만의 소량일 경우 제대로 학습이 되지 않아 정확도를 측정하기에는 한계가 있음이 확인되었다. 이에 본 논문에서는 이 한계점을 개선하고자 메모리가 추가된 신경망 모델을 활용한다.

MANN 모델의 컨트롤러 신경망은 LSTM을 사용하였고, 벡터 값을 저장할 수 있는 메모리 행렬은 128x40 크기를 사용하였다. 자세한 파라미터 값들은 Table 1. 과 같다.

Table 1. Parameters Setting

Architecture		Parameters	Value
Controller	LSTM (Long Short-Term Memory Models)	RNN Size	200
		layer number	1
		batch size	16
		sequence length	50
Memory	vector dimension	N	40
	vector size	M	128
Learning Model		input	400
		gamma	0.95
		learning rate	1e-3
		epoch	100,000

V. 실험 및 결과

본 장에서는 제안하는 모델 검증을 위해 데이터 전처리 과정과 실험에 사용한 데이터셋, 이를 이용한 실험 과정 및 결과에 대해 기술한다.

5.1 실험

5.1.1 실험 데이터

제안 모델 검증을 위해 두 가지의 데이터셋을 사용하였다. 첫 번째 데이터셋은 Kabanga 외[18], Nataraj 외[19], Agarap 외[20], Kosmidis 외[21] 등의 연구에서 활용되었던 Malimg 데이터셋이고 두 번째 데이터셋은 자체적으로 수집한 Mal60 데이터셋[22]이다.

악성코드 이미지화와 관련된 연구에서 활용되는 Malimg 데이터셋은 25개의 패밀리로 구분되어 있고, 각 패밀리들의 분류명은 Fig.5.와 같다. 데이터셋은 총 9,458 개의 악성코드 샘플들로 구성되어 있으나 본 논문에서는 소량의 샘플들을 이용해 분류하는 것이 목적이므로 각 패밀리별로 20개의 샘플만을 랜덤으로 추출하여 총 500개의 샘플로 구성하였다.

No.	Family	No.	Family
1	Allaple.L	14	Alueron.gen!J
2	Allaple.A	15	Malex.gen!J
3	Yuner.A	16	Lolyda.AT
4	Lolyda.AA 1	17	Adialer.C
5	Lolyda.AA 2	18	Wintrim.BX
6	Lolyda.AA 3	19	Dialplatform.B
7	C2Lop.P	20	Dontovo.A
8	C2Lop.gen!g	21	Obfuscator.AD
9	Instantaccess	22	Agent.FYI
10	Swizzot.gen!I	23	Autorun.K
11	Swizzor.gen!E	24	Rbot!gen
12	VB.AT	25	Skintrim.N
13	Fakerean		

Fig. 5. Malware Family in Malimg Dataset

Mal60 데이터셋은 국내 보안업체 AhnLab으로부터 제공받은 샘플을 사용하였다. 해당 데이터셋은 60개의 패밀리로 구분되어 있으며, 각 패밀리 당 20개씩 총 1,200개의 악성코드 샘플로 구성되어 있다. 진단명에 대한 업체별 정책 및 기준이 상이한 관계로, AhnLab의 진단명 외 타사의 진단명을 추가로 확인해 샘플 검증의 신뢰도를 높이고자 하였다. 이와 관련해 Fig.6.은 VirusTotal[23]에서 확인한 Kaspersky, Bitdefender, AhnLab 의 진단명을

No.	Ahnlab	Kaspersky	Bitdefender	No.	Ahnlab	Kaspersky	Bitdefender
1	Abnores	MSIL	PasswordStealerA	31	LoadMoney	GLDCT	Symmi
2	Adloader	AdLoad	-	32	Locky	Locky	Zusy
3	Adposhel	Adposhel	Razy	33	LolBot	LolBot	Generic
4	Akdoor		-	34	RansomCrypt	Locky	Agent
5	Banki	Generic	Symmi	35	MicroNames	-	Graftor
6	BitCoinMiner	Miner	Strictor	36	Nymaim	Nymaim	Agent
7	Cabonet	Generic	Crifi	37	OnlineGameHack	OnLineGames	Crypt
8	Cerber	Cerber	Generic	38	Opnumpack	Yakes	Razy
9	Cimmus	BHO	Sogou	39	OutBrowse	OutBrowse	-
10	Lukitus	Ransom	Cerber	40	OxyPumper	AdLoad	Zusy
11	CryptXXX	Generic	CryptXXX	41	Paylonjek	-	Injector
12	DLBoost	DLBoost	InstallMonster	42	Petya	Yakes	Agent
13	DNSChanger	Autorun	Kazy	43	Rifdoor	Agent	Zusy
14	Downloader	Downloader	-	44	SageCrypt	SageCrypt	Zusy
15	Encraas	Zaitu	Sarento	45	Sality	Sality	Sality
16	Farfli	Generic	Kazy	46	Spora	Spora	Razy
17	GhostRat	Runonce	Generic	47	Starter	Starter	StartPage
18	Gupboot	Generic	Symmi	48	StartSurf	StartSurf	Agent
19	HackTool	PSWTool	-	49	Sytro	Sytro	Generic
20	HDC	Generic	Symmi	50	Tesla	Androm	Cripack
21	Hebogo	-	Jaiko	51	Tiggre	Generic	Barys
22	ICLoader	Adware	ICLoader	52	Upbot	-	-
23	Infostealer	Fareit	Generic	53	Urelas	Generic	Generic
24	InstallCore	InstallMonster	-	54	VBKrypt	VBKrypt	Symmi
25	InstallMonster	Generic	InstallMonster	55	VenusLocker	Generic	Bafometos
26	Jorik	Jorik	VBKrypt	56	Vilsel	Vilsel	Generic
27	Klone	Klone	Tatef	57	WannaCrypter	Wanna	WannaCrypter
28	Korat	Generic	Generic	58	Yirith	Yirith	Generic
29	Koutodoor	Generic	Buzy	59	Zbot	Zbot	Kazy
30	Linkury	Linkury	Ursu	60	Hwdoor	-	-

Fig. 6. Malware Family in Mal60 Dataset

나타낸다.

진단명 확인 결과, 업체별로 사용하는 악성코드 패밀리 이름이 일부 다른 경우는 있었으나, 이는 명명법의 기준에 따른 차이로 동일한 그룹의 악성코드로 분류됨을 확인하였다. 예외적으로 Kaspersky와 Bitdefender 제품에서 진단명이 확인되지 않거나, 휴리스틱 진단명으로 분류되는 악성코드가 존재했는데, 해당 악성코드는 국내에서만 발견되는 Akdoor, Rifdoor 계열 샘플로서 국내 방산 업체를 타깃으로 한 악성코드 샘플로 확인되었다.

5.1.2 데이터 분류 실험 방법

본 실험은 각각 그룹에 속한 데이터들이 소량이기 때문에 학습 시, 오버피팅을 방지하기 위하여 90도, 180도, 270도 회전 시켜 데이터를 증가했다. 이렇게 증가한 데이터를 포함해 패밀리의 약 70%를 학습에 사용하였고 나머지 30%의 패밀리는 테스트에 사용하였다.

메모리가 증가된 신경망 모델을 사용하였고 모델 학습 시, 훈련 데이터는 Fig.7.과 같이 5개 패밀리에서 각각 10개의 샘플을 임의로 선택하여 총 50개로 구성한다. 약 70%의 학습 데이터에 대해 100,000 epoch 을 반복하여 학습하고 모델 검증을 위한 테스트 시에는 학습에 전혀 사용하지 않았던 패밀리에서 랜덤으로 5개를 선택한다. 선택된 5개의 샘플들에 대해 1번부터 10번까지의 횟수로 보여주었다. 보여준 횟수에 비례해 여러번 보았던 샘플을 기억하고 정확하게 분류할 수 있는 지를 테스트한다.

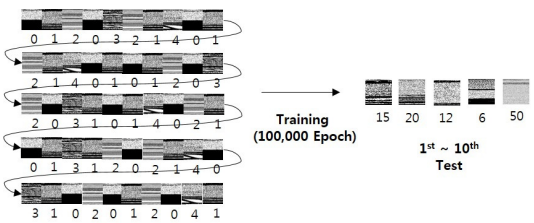

Fig. 7. Model Training & Testing

5.2 실험 결과 및 평가

실험 결과, 제안하는 모델은 Malimg 데이터셋에 대해 최대 91%의 정확도로 악성코드를 분류하였고, Mal60 데이터셋에 대해서는 최대 97% 의 정확도를 보였다. 추가로 다른 모델과의 비교를 위해Feed-Forward, CNN, LSTM 모델에 대해서도 동일한 조건에서 실험하여 MANN 모델이 다른 모델들 보다 더 좋은 성능을 보임을 확인하였다.

5.2.1 실험 결과

Malimg 데이터셋은 25개의 패밀리 중 약 70% 정도인 16개의 패밀리를 트레이닝 데이터로 사용하고, 나머지 9개의 패밀리를 테스트 데이터로 사용하였다. 자체적으로 수집한 Mal60 데이터셋은 60개의 패밀리 중 45개의 패밀리를 트레이닝 데이터로 사용하고 15개의 패밀리를 테스트 데이터로 사용하였다.

Table.2. 는 Malimg 데이터셋에서 9개의 패밀리에 대해 테스트를 해보았을 때 각 모델별 정확도를 나타낸 결과이고, Table.3. 는 Mal60 데이터셋에서 학습에 사용하지 않은 15개의 패밀리 샘플들에 대해 테스트를 해보았을 때 각 모델별 정확도를 나타낸 결과이다.

MANN 으로 학습된 모델은 처음 악성코드를 보았을 때 거의 구분을 하지 못하지만 2번째 보았을 때부터는 약 66%, 10번째 보았을 때는 약 91%까지의 정확도를 확인할 수 있다. 반면에, 별도의 메모리 구조가 없는 Feed-Forward, CNN 으로 학습된 모델은 횟수와 상관없이 높지 않은 정확도를 보여주었고 일부 기억할 수 있는 메모리가 있는 LSTM

Table 2. Malimg Dataset Accuracy

Model	Instance Accuracy (%)				
	1st	2nd	3rd	5th	10th
Feed-Forward	0.447	0.379	0.384	0.300	0.157
CNN	0.301	0.426	0.442	0.506	0.333
LSTM	0.200	0.562	0.575	0.641	0.697
MANN	0.287	0.662	0.800	0.794	0.913

Table 3. Mal60 Dataset Accuracy

Model	Instance Accuracy (%)				
	1st	2nd	3rd	5th	10th
Feed-Forward	0.312	0.290	0.333	0.300	0.321
CNN	0.409	0.340	0.285	0.250	0.222
LSTM	0.212	0.637	0.712	0.743	0.733
MANN	0.312	0.700	0.875	0.911	0.977

의 경우 상대적으로 정확도가 높았으나, MANN에 비해 낮은 정확도로 분류하였다.

5.2.2 모델 성능 평가

정확도만으로는 제안하는 모델의 성능을 평가하기 어려우므로 본 논문에서는 오차 행렬과 Precision, Recall, F1-Score 값을 이용해 모델을 평가한다.

테스트에 사용된 데이터셋은 학습에 사용되지 않았던 패밀리로 구성되어 있기 때문에 1번부터 10번까지 보여줬을 때 정확도의 차이가 있다. 이에 따라 오차 행렬은 1번부터 10번까지의 합친 값으로 계산하였고, 테스트 패밀리에서 5개씩 샘플들을 선택해 테스트 하였으므로 오차 행렬의 크기는 5 x 5 크기의 행렬이 된다.

Table. 4.와 Table.5.는 Malimg 데이터셋, Mal60 데이터셋에 대한 오차행렬을 나타낸다. 이 행렬을 이용하여 Precision, Recall, F1-Score 값을 산출하기 위해 사용한 수식은 아래와 같다.

$$Precision = \frac{(TP)}{(TP+FP)} \tag{8}$$

Table 4. Malimg Confusion Matrix (Avg. 1st ~ 10th)

	f1	f2	f3	f4	f5	total
f1	122	9	7	12	4	154
f2	6	122	10	18	2	158
f3	4	4	127	6	12	153
f4	4	15	12	132	14	177
f5	7	6	12	13	120	158
total	143	156	168	181	152	800

Table 5. Mal60 Confusion Matrix (Avg. 1st ~ 10th)

	f1	f2	f3	f4	f5	total
f1	144	7	5	3	4	163
f2	2	150	4	4	10	170
f3	7	11	128	5	12	163
f4	5	17	6	116	9	153
f5	2	8	10	2	129	151
total	160	193	153	130	164	800

Precision (8)은 제안한 모델이 분류한 결과가 실제로 맞게 분류한 비율을 나타낸다.

$$Recall = \frac{(TP)}{(TP+FN)} \tag{9}$$

Recall (9)은 각 패밀리의 악성코드들 중 제안한 모델이 맞게 분류했는지에 대한 비율을 나타낸다.

$$F1-Score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{10}$$

Recall 과 Precision 은 서로 Trade-Off 관계이므로 조화 평균인 F1-Score (10) 값을 구하여 사용한다. 이 값이 높을수록 성능이 좋음을 의미한다.

Table.6. 와 Table.7.은 제안하는 모델로 분류했을 때 각각의 실험 데이터셋에 대한 Recall, Precision, F1-Score 을 나타낸다. 이는 1번부터 10번까지 보여줬을 때의 오차 행렬을 통해 구한 값이며, Malimg 데이터셋은 평균 78%, Mal60 데이터

Table 6. Malimg Dataset Precision, Recall, F1-Score (avg. 1st ~ 10th)

	Precision	Recall	F1-Score	Total Count
f1	0.85	0.79	0.82	154
f2	0.78	0.77	0.78	158
f3	0.76	0.83	0.79	153
f4	0.73	0.75	0.74	177
f5	0.79	0.76	0.77	158
Avg. /Total	0.78	0.78	0.78	800

Table 7. Mal60 Dataset Precision, Recall, F1-Score (avg. 1st ~ 10th)

	Precision	Recall	F1-Score	Total Count
f1	0.90	0.88	0.89	163
f2	0.78	0.88	0.83	170
f3	0.84	0.79	0.81	163
f4	0.89	0.76	0.82	153
f5	0.79	0.85	0.82	151
Avg. /Total	0.84	0.83	0.83	800

셋은 평균 80% 이상의 값을 가진다.

Fig.8. 은 Malimg 데이터셋의 개수와 본 논문에서 사용한 샘플 개수의 차이를 나타낸다. 기존 Nataraj 외[19] 연구는 모든 샘플들을 사용하여 최대 97.4%의 정확도를 얻었다. 본 실험에서는 각 그룹당 20개의 샘플을 이용하였으며, 최대 91.3%의 정확도로 분류가 이뤄진 것을 확인하였다. 이 연구 결과를 통해 소량으로의 샘플로도 높은 정확도로 분류할 수 있음을 확인하였다.

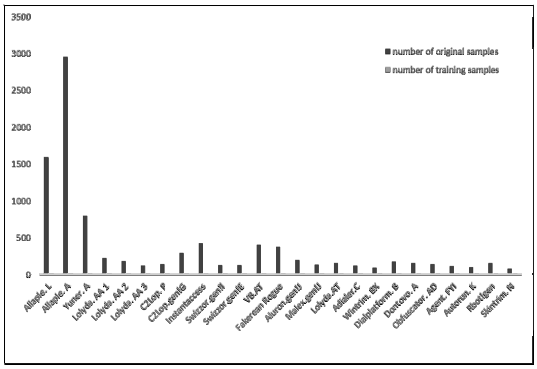

Fig. 8. Comparison of Sample Counts

VI. 결 론

본 논문에서는 타깃형 악성코드의 특징과 관련해 이미지 변환 후, 메모리가 증가된 신경망을 적용하여 샘플의 수가 충분하지 않더라도 패밀리를 분류할 수 있는 방법을 제안하였다. 사용한 데이터셋은 각 패밀리 당 20개씩 25개의 패밀리로 구성된 Malimg 데이터셋과 60개의 패밀리로 구성된 Mal60 데이터셋이다. 제안한 모델은 Malimg 데이터셋에 대해 최대 91.3% 의 정확도로 분류를 하였고 Mal60 자체 데이터셋의 경우 최대 97%의 정확도로 분류를 하였다. 이처럼 제안하는 분류 기법은 샘플의 수가 충분히 확보되지 않을 경우 정확도가 낮아지는 기존 방법의 한계점을 극복하여 소량의 샘플만으로도 높은 정확도를 가질 수 있음을 확인하였다. 이를 이용하면 APT (Advanced Persistent Threat) 와 같이 소량의 샘플만이 발견되는 악성코드들을 효과적으로 분류할 수 있기 때문에 악성코드를 프로파일링 하거나 트래킹하는 시스템에 적용할 수 있을 것이다.

하지만 계산의 복잡성을 낮추기 위한 다운사이징 과정에서 악성코드의 세부적인 특징들이 일부 사라지는 한계점이 있다. 또한 코드의 흐름 및 패턴이 극히 일부분만 유사한 경우 정확도가 낮아지게 된다. 향후에는 이 점을 보완하기 위해 악성코드를 함수 단위로 구분해 세부적인 이미지 표현하는 방법과 각 함수별 디스어셈블리 코드의 유사성을 분석하는 방향으로 추가 연구를 진행할 계획이다.

References

[1] T. H. Park, "Cyber Threat Trends Quarterly Report," Korea Internet & Security Agency, pp. 39-57, July. 2017.

[2] AV-Test, https://www.av-test.org /en/statistics/malware, Apr. 2018.

[3] AhnLab Statistics, https://www. ahnlab.com/kr/site/securityinfo/statistics/security1.do, Apr. 2018.

[4] Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S., "Malware images: visualization and automatic classification," Proceedings of the 8th international symposium on visualization for cyber security. no. 4, July. 2011.

[5] Seonhee Seok and Howon Kim, "Visualized Malware Classification Based-on Convolutional Neural Network," Journal of the Korea Institute of Information Security & Cryptology, 26(1), pp. 197-208, Feb. 2016.

[6] Gibert Llauradó, D., "Convolutional neural networks for malware classification," MS thesis. Universitat Politècnica de Catalunya, Oct. 2016.

[7] Meng, X., Shan, Z., Liu, F., Zhao, B., Han, J., Wang, H., and Wang, J., "MCSMGS: Malware Classification Model Based on Deep Learning," Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017 International Conference IEEE, pp. 272-275, 2017.

[8] Singh, A., "Malware Classification using Image Representation," Diss. INDIAN INSTITUTE OF TECHNOLOGY K

ANPUR, July. 2017.
[9] K. Han, B. Kang, and E. G. Im, "Malware Analysis Using Visualized Image Matrices," The Scientific World Journal, vol. 2014, no. 7, pp. 1-15, 2014.
[10] Kaggle, "First Place Team: Say No to Overfitting, Winner of Microsoft Malware Classification Challenge (BIG 2015)", http://blog.kaggle.com/2015/05/26/microsoft-malware-winners-interview-1st-place-no-to-overfitting, Apr. 2018
[11] T. W. Lim, "A Study on Deep Learning based Malware Detection Using Executable File Visualization and Word2Vec," Sungkyunkwan University, Aug. 2017.
[12] Huang, T. H. D., Yu, C. M., and Kao, H. Y., "R2-D2: ColoR-inspired Convolutional NeuRal Network (CNN)-based AndroiD Malware Detections," arXiv preprint arXiv:1705.04448, Dec. 2017.
[13] Graves, A., Wayne, G., and Danihelka, "Neural turing machines," arXiv preprint arXiv:1410.5401, Dec. 2014.
[14] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T., "Meta-learning with memory-augmented neural networks," International conference on machine learning, pp. 1842-1850, June. 2016.
[15] Github, "Omniglot Dataset", https://github. com/brendenlake/omniglot, Apr. 2018.
[16] Symantec, "wannacry lazarus", https://www.symantec.com/connect/blogs/wannacry-ransomware-attacks-show-strong-links-lazarus-group, Apr. 2018.
[17] Infosec, http://blog.skinfosec.com/221234742268, Apr. 2018.
[18] Espoir K. Kamundala and Chang Hoon Kim, "CNN Model to Classify Malware Using Image Feature," KIISE Transactions on Computing Practices, 24(5), pp. 256-261, May. 2018.
[19] Nataraj, L., and Manjunath, B. S., "SPAM: Signal Processing to Analyze Malware," IEEE Signal Process. Magazine, vol. 33, no. 2, pp. 105-117, Mar. 2016.
[20] Agarap, A. F., and Pepito, F. J. H., "Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification," arXiv preprint arXiv:1801.00318, Dec. 2017.
[21] Kosmidis, K., and Kalloniatis, C., "Machine Learning and Images for Malware Detection and Classification," Proceedings of the 21st Pan-Hellenic Conference on Informatics. no. 5, Sep. 2017.
[22] Github, "Mal60 Dataset", http://github.com/pukekaka/mal60, Apr. 2018.
[23] VirusTotal, "Virus Total", http://virustotal.com, Apr. 2018.

〈저자소개〉

강 민 철 (Min Chul Kang) 정회원
2012년 2월: 단국대학교 컴퓨터공학과 졸업
2016년 9월~현재: 고려대학교 정보보호대학원 정보보호학과 석사과정
〈관심분야〉 악성코드, 머신러닝, 네트워크 보안

김 휘 강 (Huy Kang Kim) 종신회원
1998년 2월: KAIST 산업경영학과 학사
2000년 2월: KAIST 산업공학과 석사
2009년 2월: KAIST 산업 및 시스템공학과 박사
2004년 5월~2010년 2월: 엔씨소프트 정보보안실장. Technical Director
2010년 3월~2014년 12월 고려대학교 정보보호대학원 조교수
2015년 1월~현재 고려대학교 정보보호대학원 부교수
〈관심분야〉 온라인게임 보안, 네트워크 보안, 네트워크 포렌식