Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.1, Feb. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.1.197

Convolutional Neural Network 기반의 악성코드 이미지화를 통한 패밀리 분류*

석 선 희,† 김 호 원‡
부산대학교

Visualized Malware Classification Based-on Convolutional Neural Network*

Seonhee Seok,† Howon Kim‡
Pusan National University

요 약

본 논문에서는 악성코드를 실행시키지 않고 패밀리를 분류하는 방법으로 악성 코드 파일을 8-bit gray-scale 이미지로 시각화 하고 이미지 인식분야에서 널리 쓰이고 있는 convolutional neural network를 통해 악성코드를 분류해내는 기법을 제안한다. 9개의 악성코드 패밀리로 분류해 내는 실험의 Top-1,2 예측 정확도는 각각 96.2%, 98.7%을 기록하였고, 27개의 패밀리를 분류하는 실험의 경우 Top-1 예측 정확도는 82.9%, Top-2는 89%로 악성코드 패밀리를 분류할 수 있다.

ABSTRACT

In this paper, we propose a method based on a convolutional neural network which is one of the deep neural network. So, we convert a malware code to malware image and train the convolutional neural network. In experiment with classify 9-families, the proposed method records a 96.2%, 98.7% of top-1, 2 error rate. And our model can classify 27 families with 82.9%, 89% of top-1,2 error rate.

Keywords: Malware Classification, Malware Image, Convolutional Neural Network

I. 서 론

AV-Test 의 조사[1]에 따르면 2014년 7월부터 2015년 6월까지 매달 약 13,000,000개의 새로운 악성코드가 만들어지고 있으며 시간이 지날수록 새로운 악성코드의 숫자는 증가하는 추세이다. 매일 쏟아져 나오는 다양한 유형의 악의적인 프로그램들에 대해 올바른 대처 방법을 제시하기 위해서는 악성코드 패밀리의 정확한 분류가 선행되어야 하고 이에 따라 적절한 방어기법이 제공되어야 한다. 이를 위해 악성코드 패밀리를 분류하기 위한 다양한 방법들이 연구되고 있는데 분류 성능을 높이기 위해서는 패밀리를 구분 짓는 악성코드 특징에 대해 연구하거나 분류기 자체의 성능을 높이는 두 가지 방법이 존재한다.

악성코드들마다 가지고 있는 고유한 특성을 찾는 방법은 악성코드를 실행시키지 않고 악성코드 분석만을 이용하여 명령어 사용빈도, 함수 호출 관계, 코드상의 String 정보를 분석하는 정적 분석 기법[2, 3, 4, 5]과 악성코드를 가상의 환경에서 실행시켜 행동 패턴을 분석하는 정적 분석 기법[6, 7] 또는 이 두 가지를 모두 이용하는 방법[8, 9]이 존재한다.

악성코드 패밀리를 분류할 때 악성코드의 고유특

Received(12. 10. 2015), Modified(01. 13. 2016), Accepted(01. 14. 2016)
* 본 연구는 2015년도 산업통상자원부의 재원으로 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구 과제입니다. (No. 20152000000170)
† 주저자, seokseonhee@gmail.com
‡ 교신저자, howonkim@pusan.ac.kr(Corresponding author)

징을 잘 찾아내는 것도 중요하지만 분류기로 사용되는 분류 알고리즘을 올바르게 선택하는 것도 중요하다. 최근 분류 혹은 인식기법을 연구하는 분야에서 가장 활발하게 연구되고 있는 분야는 신경망의 은닉계층의 수를 늘려 만든 심층 신경망이라고 하는 deep neural network (DNN) 관련 연구 분야일 것이다. 특히 이미지와 음성 인식분야에서 심층 신경망기반의 모델들이 뛰어난 성능을 기록하면서 이를 다른 분야에도 이용하려는 움직임들이 나타나는데 악성코드 분석 역시 그러한 분야 중 하나이다. 실제로 심층 신경망을 이용한 다양한 악성코드분류 모델들이 제시되고 있는데 Recurrent Neural Network (RNN) 를 이용한 분류 기법[10]과 RNN와 deep feed-forward neural network이라는 서로 다른 두 가지의 심층 신경망을 이용한 악성코드 분류 연구[11]가 그 예이다.

본 논문에서 제안하는 모델은 악성코드의 추가적인 분석 없이 악성코드를 이미지화한 것을 이용하여 분류하는 방법이다. 또한 분류기로는 다양한 심층 신경망 모델 중 이미지 인식분야에서 뛰어난 성능을 보이는 Convolutional Neural Network(CNN)을 이용한다. CNN에 사용되는 convolution 연산이 이미지의 회전, 변형, 경계선 검출에 장점을 가진다는 점과 악성코드파일을 이미지화 하였을 때 고유한 특징이 나타남을 이용하여 악성코드 패밀리를 분류하는 기법을 제안한다.

본 논문의 구성은 2장에서 악성코드 분류기법의 관련연구에 관해 서술하고 3장에서는 제안하는 모델에 대해 상세 기술한다. 4장에서는 제안하는 모델을 이용한 실험에 대해 기술하고 5장에서는 실험 결과에 대해 분석하며 6장에서 결론과 향후 연구 방향에 대해 제시한다.

II. 관련 연구

악성코드는 코드 변형을 통해 많은 변이들을 생성하게 되는데 악성코드 내용를 이용하여 생성한 시그니처 기반의 분류 모델은 사전 정의 되지 않은 시그니처가 입력으로 들어온 경우 분류가 어렵기 때문에 악성코드 패밀리 변이들을 분류해 내기 어렵다는 문제를 가지고 있다. 따라서 이 문제를 해결하기 위해 M. Bailey[6]는 가상의 환경에서 악성코드를 실행시키고 이 때 변화 되는 운영체제의 메모리나 레지스터등을 추적하여 악성코드의 행동에 대한 fingerprint를 생성하여 패밀리를 분류한다. 하지만 이 방법을 통해 악성코드를 분류해내기 위해서는 모든 악성코드를 실행해봐야 하므로 사용 편의성에 제약을 가진다.

W. Jung[11]의 연구에서는 zero-day flash 악성코드 탐지를 위해 심층 신경망 모델을 이용하였다. Adobe flash 제품에 대해 제조사에서 취약점 패치를 배포하기 전에 해당 취약점을 공격하는 zero-day flash 악성코드를 탐지하기 위한 방법으로 악성코드로부터 추출한 많은 수의 특징점들을 효율적으로 학습하기 위해서 기존의 신경망 모델에서 은닉계층의 숫자를 증가시킨 심층 신경망을 이용하여 생성한 모델과 순차적인 데이터 분석에 유리한 심층 신경망의 변형 버전인 RNN을 이용하여 악성코드 탐지 모델을 만들었다. 해당 연구 역시 많은 악성코드 특징점을 이용하여 패밀리를 분류하기에 높은 정확도로 분류해 낼 수 있으나 하나의 악성코드의 특징점을 추출하는데 너무 오랜 시간이 걸린다는 단점을 가진다.

L. Nataraj[12]의 연구에서는 악성코드 파일을 시각화하여 분류하는 방법을 제안하였으며 분류기를 학습시키기 위한 이미지 gabor filter를 이미지 특징점 추출기로 이용하고 k-Nearest Neighbors (kNN)를 분류기로 이용하였다. Nataraj의 모델은 앞의 두 기법들에 비해 간단한 방법으로 악성코드의 유형을 분류 해 낼 수 있지만 악성코드를 이미지화 시키고, 변환된 이미지를 기존의 기계 학습 알고리즘에 학습시키기 위해 특징점 추출기를 추가로 배치해야한다는 번거로움이 존재한다.

본 논문에서 제안하는 모델은 악성코드를 이미지화 한 후 CNN을 이용하여 악성코드 패밀리를 분류하는 모델을 제안한다. 기존의 악성코드 유형 분류 모델에서 코드내용을 상세히 분석하거나 실행시켜 행동을 분석하는 방법을 사용하는 것과 달리 악성코드를 이미지로 변환시키는 과정만을 통하여 악성코드를 분류해 낼 수 있으며, 악성코드 이미지를 이용할 때 기존의 다른 연구들이 이미지 특징점 추출기와 분류기를 따로 사용한 것에 비해 제안하는 기법은 CNN만을 이용하기 때문에 전체 악성코드 분류과정을 간소화 할 수 있다.

III. 제안 모델

제안하는 악성코드 패밀리 분류 기법은 이미지 인

식 분야에서 뛰어난 성능을 보이는 CNN를 이용하여 시각화한 악성코드의 패밀리를 분류해내는 방법이다.

CNN은 입력 데이터로 이미지를 사용하므로 악성코드 바이너리 코드를 이미지화 하는 과정이 필요하고, 이 악성코드 이미지를 CNN에 학습시켜 악성코드 패밀리 분류기로 사용할 수 있다.

제안하는 악성코드 분류기는 크게 두 부분으로 구성 되는데 입력되는 악성코드 바이너리 파일을 이미지로 변환시키는 악성코드 이미지 변환기와 변환된 악성코드 이미지를 입력으로 패밀리를 예측하는 CNN으로 구성된다. CNN을 악성코드 분류기로 사용하기 위해서는 CNN의 가중치 값을 학습 시키는 과정이 필요하므로, 학습데이터 샘플들을 악성코드 이미지 파일과 악성코드 패밀리의 쌍으로 구성하여 CNN을 학습한다. 학습된 CNN에 악성코드 이미지 변환기로부터 출력되는 악성코드 이미지를 입력하면 학습된 CNN의 가중치 값들과 연산을 수행한 후 패밀리별로 예측 확률을 결과 값으로 출력해준다.

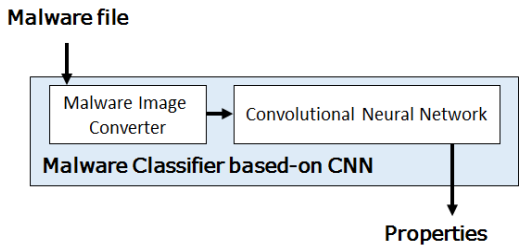

Fig. 1. Architecture of Proposed Model

3.1 악성코드 이미지 생성

제안하는 악성코드 분류 시스템에서는 악성코드 이미지 변환기를 이용하여 악성코드 파일을 이미지로 변환 시키는 작업을 가장 먼저 수행한다. 악성코드를 이미지로 변환시키는 과정에 대해 상세히 알아보자.

제안하는 방법에서 사용하는 CNN기반의 악성코드 패밀리 분류기는 256x256 (65536 byte) 크기의 이미지를 입력으로 사용하기 때문에 악성코드 이미지를 일정한 크기로 변환하는 과정이 필요하다.

따라서 악성코드 파일을 정사각형 모양의 이미지로 변환시키고 분류기의 입력 이미지 형태인 256x256 크기의 8-bit gray-scale 이미지가 되도록 축소 또는 확대한다.

먼저 악성코드를 정사각형의 이미지로 변환시키는

```
Algorithm : Malware Image Conversion
# Variables
m : malware byte array, s : malware file size,
img : malware Image , n : malware image size

# Step 1 : Convert squared malware image
n = Ceil(√s)
if n^2 > s :
  m = AddZero(m)
img = SaveImage(m, n)

# Step 2 : Resize image
(m_img : malware Image)
if n > 256 :
  img = DownSample(img)
if n < 256 :
 img = UpSample(img)
```

Fig. 2. Algorithm for Malware Image Conversion

과정에 대해 상세히 알아보자. 예를 들어 65536 byte이상의 크기를 가지는 악성코드 파일을 정사각형 모양의 이미지로 변환시키기 위해 이미지의 한 변의 길이를 구하려면 전체 악성코드 파일 크기의 root 연산 값에 ceil 함수를 적용한 결과 값을 구한다. 즉, 악성코드 파일 크기를 S 악성코드 이미지의 한 변의 길이 W라 할 때, W는 수식 (1) 과 같다.

$$W = \lceil \sqrt{S} \rceil \quad (1)$$

따라서 악성코드 이미지는 $W \times W$ 크기의 이미지로 변환되며 $S < W^2$인 경우 부족한 byte는 0으로 채운다. 악성코드 파일의 크기가 65536 byte보다 작은 경우에도 수식 (1)을 이용하여 위와 동일한 방법으로 악성코드 이미지의 한 변의 길이를 구하여 악성코드 파일을 이미지로 변환할 수 있다.

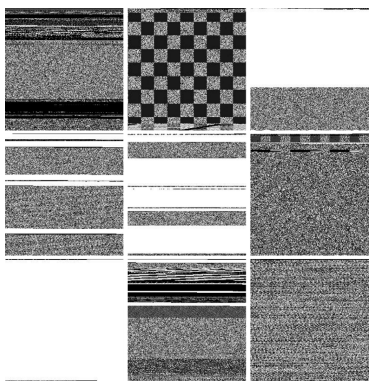

Fig. 3. Example Malware Image of Microsoft Dataset

이렇게 변환시킨 다양한 크기의 악성코드 이미지는 분류기입력을 위한 고정된 크기를 조정하기 위해 악성코드 이미지를 확대 또는 축소시켜 악성코드 이미지 변환 단계를 마친다. Fig.3는 Microsoft 데이터 셋에 속하는 9개의 패밀리 별 악성코드 샘플을 이미지로 변환시킨 예이다.

3.2 악성코드 패밀리 분류기 설정

제안하는 악성코드 분류 모델에서 사용하는 분류기는 CNN으로 분류기의 주요 연산은 convolution 연산이다. 따라서 많은 양의 데이터를 학습이 필요한 CNN의 경우 convolution 연산을 효율적으로 처리할 수 있어야 모델을 학습하는데 시간이 적게 걸리는데, 이를 위해서 GPU를 이용하여 convolution 연산을 병렬적으로 처리하는 방법을 이용한다. 이러 CNN모델의 특성을 반영하여 GPU기반의 딥 러닝 알고리즘을 구현한 오픈 소스 형태의 프레임워크들이 많이 존재한다. 제안하는 모델을 위하 사용한 프레임워크는 최초로 이미지 분류 분야에 성공적으로 CNN 기법을 적용한 Alexnet[13]을 기반으로 하는 Cuda-Convnet2 프레임워크를 이용한다. Cuda-Convnet2는 CNN을 구성하는 기본적인 layer과 layer 파라미터 등을 자유롭게 설정할 수 있으며 두 개 이상의 GPU를 이용하여 빠른 CNN 알고리즘 수행이 가능하도록 기능이 제공된다.

본 제안 시스템에서 이미지화 된 악성코드를 분류해 내기 위해 구성한 CNN은 4개의 convolution layer와 2개의 full connect layer 2개로 구성되며 convolution layer 사이에 pooling layer를 배치하여 convolution layer의 output 숫자, 즉 특징점의 차원을 줄이도록 하였다. 이전 layer의 모든 출력 값을 신경망 형태로 연결한 full connect layer사이에는 dropout layer를 배치하여 training data 의 과적합을 방지한다. 활성함수로는 $f(x)=\max(0,x)$인 Rectifier Unit(Relu)를 사용하였다. 또한 256x256크기의 이미지를 임의의 위치에서 224x224 크기로 잘라 입력으로 사용하는 input layer를 사용하여 유사한 이미지라 하더라도 다양한 모양으로 학습될 수 있도록 한다.

Fig.4은 제안하는 모델을 이용하여 Microsoft 데이터 셋의 악성코드분류를 위해 구성한 CNN의 구조이다. 분류기의 상세 layer 및 파라미터는 Table 1과 같다.

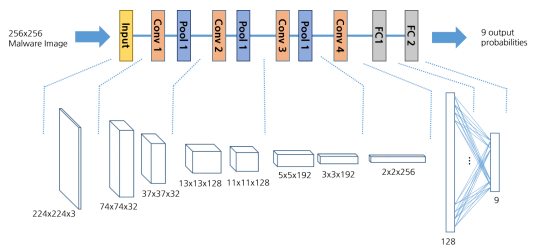

Fig. 4. The CNN architecture for proposed model

Table 1. The parameters of proposed CNN

layer	parameters	value	output
Input		256x256	224x224
Conv1	n of filter	32	74x74x32
	filter size	5x5	-
	filter stride	3	-
	channel	3	-
Pool1	filter size	3x3	37x37x32
	filter stride	2	
Conv2	n of filter	128	13x13x128
	filter size	5x5	-
	filter stride	3	-
	channel	32	-
Pool2	filter size	3x3	11x11x128
	filter stride	1	
Conv3	n of filter	192	5x5x192
	filter size	5x5	-
	filter stride	2	-
	channel	128	-
Pool3	filter size	3x3	3x3x192
	filter stride	1	
Conv4	n of filter	256	2x2x256
	filter size	3x3	-
	filter stride	2	-
	channel	192	-
FC1			128
FC2			9

Input layer에서는 256x256 크기의 이미지를 입력으로 하고 입력된 이미지를 임의의 위치에 224x224 크기로 잘라 CNN의 첫 번째 convolution layer인 Conv1 layer의 입력으로 사용한다. Convolution layer의 경우 convolution filter의 숫자와 크기, convolution 연산 적용 간격과 data의 channel을 정의해야 하면 출력 값이 결정되고, pooling layer는 pooling filter 크기와 간격을 정의하면 출력 값이 결정된다. Full-connect layer는 FC1의 경우 마지막 convolution layer인 Conv4에 출력 값인 1024개

의 데이터를 입력 값으로 하고 출력 값은 128개로 한다. FC2는 Microsoft 데이터 셋의 패밀리 유형의 개수인 9를 출력 값 개수로 정한다.

IV. 실 험

본 장에서는 제안하는 모델의 패밀리 분류 성능을 검증하기 위해 두 가지 데이터 셋을 이용하여 패밀리 종류에 따른 분류 성능 실험에 대해 기술한다.

4.1 실험 환경

실험을 위한 환경은 Linux Mint 17.1 Rebecca - cinnamon(64bit) 운영체제에서 Convnet2 프레임워크와 CUDA 7.0를 이용하여 실험하였다. 전체 상세 실험 환경은 Table 2에 기술되어 있다.

Table 2. Experiments environments

Name	Spec.
OS	Linux Mint 17.1(64bit)
CPU	Inter Core i5-4670 3.40GHz
RAM	8G
GPU	GTX 780
Cuda	7.0

4.2 실험 방법

제안하는 모델의 성능을 검증하기 위해 두 가지 실험을 진행한다. 첫 번째는 주어진 악성코드를 9개의 패밀리 중 하나로 분류하는 실험이며, 두 번째 실험은 악성코드 패밀리의 숫자를 늘려 패밀리의 숫자가 늘어났을 때에도 제안하는 모델이 적절히 동작하는지 알아보기 위한 실험이다. Fig.5는 실험과정을 표현한 내용이다.

두 가지 실험은 모두 10-fold cross validation 기법을 적용하여 모델을 학습하고 검증하는 과정 거친다. 즉, 데이터 셋을 10등분하여 성능측정 실험을 10번 반복한 후 평균을 구하여 모델의 실험 결과를 산출하며 실험 결과는 top-1,2 오류율로 표시된다. 분류기는 악성코드분류결과를 학습된 모든 패밀리에 대해 각각의 패밀리로 분류될 확률 값을 출력한다. 이 때 top-k 오류율이란 분류기가 예상하는 패밀리 확률 값 중 상위 k개에 악성코드 실제 패밀리가 없을 경우 오류라고 정의한다.

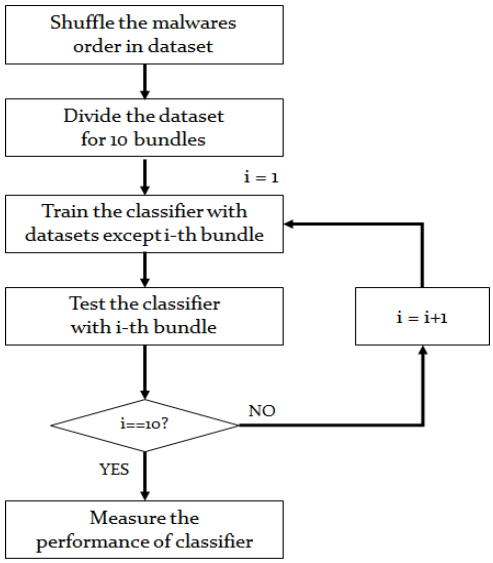

Fig. 5. The experiments process

4.3 Dataset

제안된 시스템의 분류 성능 실험을 위해 Microsoft Malware Classification Challenge (BIG 2015) 데이터 셋[14]과 VXHeaven의 2010 virus collection 데이터 셋[15]을 사용하였다.

Microsoft 데이터 셋은 9개의 패밀리 중 하나에 속하는 악성코드 파일들이 존재하며, 패밀리 레이블이 존재하는 10868개의 training set과 레이블이 존재하지 않는 10873개의 testing set으로 나눠져 있다. Microsoft 데이터 셋은 하나의 악성코드에 대해 PE 헤더가 제거된 바이트 코드 파일과 meta data manifest 파일로 제공되는데 본 논문에서는 바이트 코드 파일 만을 사용하였다.

VXHeaven 데이터 셋의 경우 270k 개의 악성코드 실행파일로 구성되어 있으나 악성코드 패밀리 정보는 포함되어 있지 않다. VXHeaven dataset 샘플의 패밀리를 알아내기 위해 악성코드 유형분석 사이트인 Virustotal[16]을 통해 샘플들에 대한 패밀리 정보를 레이블링 한다. Virustotal 사이트는 악성코드 의심 파일을 업로드하면 Anti-virus 엔진 유형별로 악성코드 패밀리 예측 결과를 분석하여 주는데, 분석 결과 중 Microsoft Anti-virus 엔진의 결과를 기준으로 악성코드의 패밀리 label을 정하고 300개 이상의 샘플을 가지고 있는 27개의 패밀리에

속하는 샘플들을 분류실험에 이용하였다.

4.4 실험 1. Microsoft Dataset 의 분류 실험

제안하는 악성코드 패밀리 분류기의 성능을 실험하기 위해 9개의 유형으로 이루어진 Microsoft 데이터 셋을 이용하였다.

이 실험에 사용하는 dataset의 샘플 수는 약 20,000개로 10-fold cross validation기법을 적용하기 위해 매 실험 시 약 18,000의 샘플은 모델을 학습하는데 사용하고 약 2,000개는 모델을 검증하는데 사용하였다. 이에 따른 실험 결과는 Table 3과 같다. 실험1 수행 결과 평균 top-1, top-2 오류율은 각각 0.038101, 0.012993를 기록하여 제안하는 모델의 경우 96.2%의 정확도로 악성코드 패밀리를 분류해 낼 수 있으며 분류기가 예측한 2개의 패밀리 중 정답이 있을 확률은 98.7%가 된다는 것을 알 수 있다.

Fig. 6은 임의로 선택된 10개의 악성코드에 대한 유형 분류결과를 나타내는데, 악성코드 이미지와 이미지 아랫부분에 실제 악성코드 유형이 표시되고, 악성코드 이름 하단 부분 박스에 분류기가 예측한 유형 중 확률 값이 높은 순서로 2개의 악성코드 이름과 확률 값이 두 개의 막대 그래프 형태로 나타나있다. 막대그래프의 색깔은 예측한 악성코드 유형이 실제 유형과 동일하면 붉은색으로, 다른 경우 파란색으로 표시되며 막대 그래프의 크기는 해당 유형일 확률을 의미 한다. Fig. 6의 하단 좌측에서 1번째의 분류결과를 보면 실제 악성코드의 유형은 GaTak 인데, 분류기가 예측한 두 개의 악성코드 이름은 Lollipop과 GaTak으로 가장 높은 확률로 예측한 Lollipop이 잘못 예측되어 파란색 막대그래프로 표

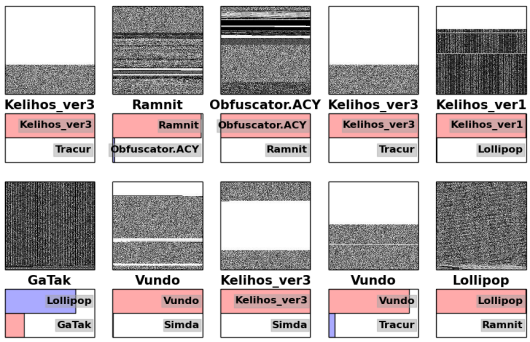

Fig. 6. The Classification results of randomly selected Microsoft samples

시되어있다.

4.5 실험 2. VXHeavens Dataset의 분류 실험

실험 1은 9개의 악성코드 패밀리를 분류하는 문제를 해결하기 위한 실험이었다면 실험 2에서는 VXHeaven 데이터 셋을 이용하여 제안하는 모델이 많은 수의 분류 클래스(악성코드 유형)가 존재할 때도 이들을 분류해 낼 수 있는지 확인하기 위한 실험을 수행한다. 실험 2는 실험 1과 동일하게 10-fold cross validation기법으로 모델의 분류 성능을 검증한다. 실험 2에서 제안하는 모델의 평균 top-1 오류율은 0.170526를 기록하여 82.9 %의 정확도로 악성코드의 패밀리 유형을 구분해 냈고, 평균 top-2 오류율은 0.109896을 기록하여 분류기가 예측한 두 개의 패밀리 중에 정답이 있을 확률은 89%를 기록하였다.

실험에 사용한 테스트 데이터 셋 중 10개의 악성코드 샘플을 임의로 선택하여 이미지화하고 분류기를

Table 3. The results of experiment 1

Exp. ID	Top-1 error	Top-2 error
Exp.1_01	0.034806	0.013833
Exp.1_02	0.028000	0.010000
Exp.1_03	0.039200	0.012000
Exp.1_04	0.040000	0.013600
Exp.1_05	0.038500	0.014000
Exp.1_06	0.040500	0.012500
Exp.1_07	0.046500	0.016500
Exp.1_08	0.031500	0.011000
Exp.1_09	0.042000	0.014500
Exp.1_10	0.040000	0.012000
Average	0.038101	0.012993

Table 4. The results of experiment 2

Exp. ID	Top-1 error	Top-2 error
Exp.2_01	0.175597	0.108457
Exp.2_02	0.176000	0.111500
Exp.2_03	0.174000	0.114000
Exp.2_04	0.165500	0.104500
Exp.2_05	0.160500	0.098500
Exp.2_06	0.154500	0.099000
Exp.2_07	0.192500	0.121000
Exp.2_08	0.166667	0.106000
Exp.2_09	0.165333	0.114667
Exp.2_10	0.174667	0.121333
Average	0.170526	0.109896

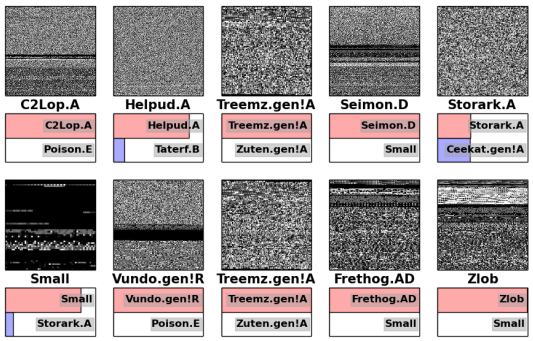

Fig. 7. The Classification results of randomly selected VXHeavens samples

통한 악성코드 유형 분류 결과를 표시하면 Fig.7 과 같이 나타난다.

Fig. 7에서 보이는 10개의 샘플 모두 분류기가 악성코드 유형을 올바르게 예측하였고, 상단 우측에서 첫 번째 샘플의 경우 분류기가 예측한 두 개의 유형에 대한 확률 값이 비슷하게 나왔으나 근소한 차이로 Storark.A 유형을 올바르게 예측하였다.

V. 분 석

이 장에서는 제안하는 모델의 분류성능에 영향을 미치는 요소를 알아보기 위해 오차 행렬을 이용하여 악성코드 유형별 분류 결과를 분석해본다. 또한 다른 악성코드 분류 모델들과의 비교를 통해 제안하는 악성코드 분류 모델이 가지는 특징을 알아보자.

5.1 오차행렬을 이용한 악성코드 분류결과 분석

머신 러닝 또는 딥 러닝 분야에서 제안하는 알고리즘의 성능에 대한 평가는 단순히 분류 정확도(Accuracy)만을 가지고 평가하기 어렵다. 따라서 분류기의 성능을 측정하기 위해서는 오차 행렬이라 하는 confusion matrix를 이용하여 분류기의 예측값과 실제 값 사이에 관계를 세분화하여 성능 지표를 정의[17]하는데, Accuracy 뿐 아니라, Recall, Precision, F-score와 같이 여러 가지 지표로 표현할 수 있다.

성능 지표를 도출하기 위해서 9개의 악성코드 유형으로 분류하는 실험 1의 결과를 오차 행렬로 다시 나타내면 Table 5과 같이 표현된다.

Table 5의 가로축에 표시된 레이블은 분류기가 예측하는 악성코드 패밀리 ID를 의미하고 세로축에 표시된 레이블은 악성코드의 실제 악성코드 패밀리 ID를 의미한다.

Table 5. The confusion matrix for exp.1

	M1	M2	M3	M4	M5	M6	M7	M8	M9	합계
M1	257	4	0	0	0	1	2	6	0	270
M2	6	406	0	0	0	9	0	4	12	437
M3	0	0	479	0	0	0	0	0	0	479
M4	0	0	2	75	0	4	0	0	0	81
M5	1	0	0	3	1	0	2	0	0	7
M6	6	0	2	1	0	107	1	3	0	120
M7	1	3	0	0	0	1	77	0	0	82
M8	11	2	4	6	0	3	0	191	0	217
M9	2	16	0	2	0	4	1	3	147	175
합계	284	431	487	87	1	129	83	207	159	1868

이 오차 행렬를 이용하여 precision, recall, f-score를 산출 할 수 있는데, 실험 1은 분류 클래스는 9개인 멀티 클래스 분류 문제이므로, 각각의 클래스 별로 해당 값을 계산한 뒤 평균값을 구하여 모델 전체의 성능을 계산 할 수 있다. 예를 들어 M2 유형의 Precision과 Recall, F-score를 구하는 과정을 살펴보면 M2의 True Positive 샘플의 수는 424이고, False Negative 샘플의 합은 M2 행의 True Positive 샘플을 제외한 나머지 원소의 합이므로 20이 되며 False Positive 샘플의 합은 M2열의 합계에서 True Positive 샘플의 수를 제외하면 10이 된다. 따라서 Precision은 $\mathrm{Pr}esicion = \frac{(TP)}{(TP+FP)} = \frac{(424)}{(424+10)}$ 로 0.976959 가 되며 Recall은 $Recall = \frac{(TP)}{(TP+FN)} = \frac{(424)}{(424+20)}$ 로 0.954955 이며 F-Score는 이 두 값의 조화 평균인 0.965831이 된다. 이와 같은 방법으로 M1부터 M9

Table 6. The results analysis of experiment 2

Family ID	Precision	Recall	F-Score
M1	0.958042	0.958042	0.958042
M2	0.976959	0.954955	0.965831
M3	0.982079	0.998179	0.990063
M4	0.956522	0.946237	0.951351
M5	0.666667	0.666667	0.666667
M6	0.941606	0.941606	0.941606
M7	0.890244	0.960526	0.924051
M8	0.959596	0.913462	0.935961
M9	0.922705	0.950249	0.936275
평균	0.917158	0.921102	0.919126

까지의 각각의 악성코드 유형별 분류 결과를 이용하여 precision과 recall, f-score를 구할 수 있다. 최종적으로 제안하는 악성코드 유형 분류 모델의 precision은 0.917158, recall은 0.921102, f-score는 0.919126이 됨을 알 수 있다.

실험 2의 분류 결과 역시 오차 행렬을 이용하여 표현하면 Table 7와 같다. V1에서 V27까지 recall과 precision의 조화 평균인 f-score를 살펴보면 0.341463에서 1.0까지 다양한 분포로 이루어져 있음을 확인 할 수 있다.

실험 1과 실험 2의 오차행렬을 살펴보면 특정 유형에 대해 F-score가 낮게 측정됨을 알 수 있다. 실험 1의 M5 유형의 F-score가 0.666667로 다른 모든 악성코드 유형들의 F-score가 0.9이상인 것에 비에 현저히 낮고 실험 2의 경우에도 V1, V6, V8, V15, V25와 같이 특정 유형의 F-Score가 0.6 이하임을 알 수 있다. 따라서 악성코드 분류 정확도에 영향을 미치는 요인을 알아보기 위해 악성코드 샘플의 수와 분류 정확도의 관계에 대해 더 알아본다.

Table 7. The results analysis of exp.2

family ID	Precision	Recall	F-Score
V1	0.583333	0.241379	0.341463
V2	0.714286	0.789474	0.750000
V3	0.805556	0.707317	0.753247
V4	0.871795	0.871795	0.871795
V5	0.873684	0.902174	0.887701
V6	0.421569	0.741379	0.537500
V7	0.950000	0.612903	0.745098
V8	0.705882	0.444444	0.545455
V9	0.787671	0.927419	0.851852
V10	0.930233	0.975610	0.952381
V11	0.743590	0.743590	0.743590
V12	0.968750	0.939394	0.953846
V13	0.989691	1.000000	0.994819
V14	0.588235	0.735294	0.653595
V15	0.571429	0.413793	0.480000
V16	0.851064	0.842105	0.846561
V17	0.615385	0.615385	0.615385
V18	0.866667	0.855263	0.860927
V19	1.000000	0.870968	0.931034
V20	1.000000	0.988636	0.994286
V21	0.787879	0.684211	0.732394
V22	0.952381	1.000000	0.975610
V23	0.888889	0.677966	0.769231
V24	0.977941	1.000000	0.988848
V25	0.500000	0.192308	0.277778
V26	1.000000	1.000000	1.000000
V27	0.871795	1.000000	0.931507
평균	0.808063	0.769363	0.788238

5.2 악성코드 샘플 수와 분류 성능과의 관계

악성코드 샘플의 수와 분류 정확도와의 관계를 확인하기 위해 악성코드 유형별 F-score와 샘플 수와의 관계를 그래프로 표현하여보자.

실험 1의 경우에는 M5의 샘플의 수는 6개이고 F-score는 0.667를 기록하였다. M4와 M7의 경우에도 각각 93개, 76개의 샘플로 구성되어 있지만 F-score는 0.951과 0.924로 상대적으로 M5 유형에 비해 많은 샘플의 수를 가지고 있으므로 너무 적은 수의 샘플로 구성된 악성코드 유형의 경우 분류 정확도에 영향을 미친다는 것을 알 수 있다.

실험 2의 경우를 살펴보면 또 다른 특징을 알 수 있다. 실험 2의 경우 실험 1과 비교하였을 때 샘플의 수가 19~133개로 비교적 고르게 분포해있으며 V26의 경우 24라는 적은 샘플의 수에도 불구하고 F-score는 1을 기록하였다. 샘플의 수가 적은 경우에 F-score가 높게나오는 경우도 있고 낮게 나오는 경우도 있으나 샘플의 수가 70개 이상인 V5, V9, V13, V16, V18, V20, V24, V27의 경우에는 모

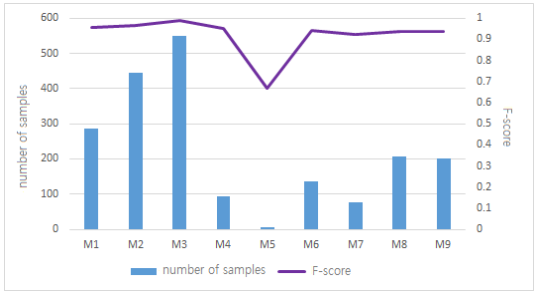

Fig. 8. The exp.1 result graph for the number of samples and F-score

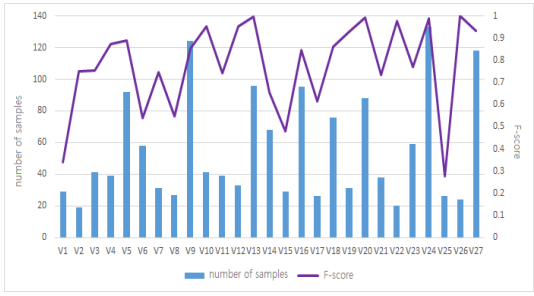

Fig. 9. The exp.2 result graph for the number of samples and F-score

두 F-score가 0.847이상을 기록하였다. 이를 미루어보아 적은 샘플의 수가 악성코드 분류모델의 성능에 절대적인 영향을 미칠 수는 없지만 악성코드 수가 일정량 이상 확보되어 CNN이 충분히 학습이 되어야 악성코드 유형을 분류해 낼 수 있음을 알 수 있다.

5.3 결과 비교

제안하는 모델과 성능을 비교할 모델은 Nataraj[12, 20]의 분류 모델, Rieck[18]의 분류 기법 , Han[19]의 분류 방법이다. 각각의 모델에 대해 간략히 소개 하면 아래와 같다.

Nataraj의 분류 모델은 제안하는 모델과 유사하게 악성코드를 시각화하여 악성코드를 분류하는데, 악성코드 이미지의 특징점을 추출하기 위해 Gabor filter를 이용하였고 추출한 특징점은 kNN (k-Nearest Neighbors) 알고리즘을 사용하여 악성코드 유형을 분류 하였다. Rieck의 분류 기법은 sandbox 환경에서 악성코드를 실행시켜 악성코드의 행동 패턴을 모니터링 한 결과를 특징점으로 삼고 SVM을 이용하여 악성코드 유형을 분류 하였다. Han의 방법은 악성코드에 사용된 API뿐 아니라 Hash 값, AV 테스트 결과 값, Packer등의 다양한 특성인자를 통해 악성코드 DNA를 생성하고 유사도 비교를 통해 악성코드를 분류하는 기법을 제안하였다. 위의 모델들과 제안하는 악성코드 분류 기법은 분류방법, 데이터 셋의 종류, 악성코드 유형의 수, 샘플의 수를 비교한다. 성능지표는 각 논문에서 가장 많이 사용된 정확도를 기준으로 사용하였으며 제안하는 악성코드 분류 기법과 이전 연구결과를 비교하여 요약하면 Table 8과 같다.

Rieck의 분류 모델은 14개의 패밀리를 88%의 확률로 악성코드 패밀리를 분류하는데 성공했고,

Table 8. Result comparisons

Model	Method	family number	accuracy
Nataraj	Gabor Filter + kNN	6	0.9514
		531	0.7280
Rieck	SVM	14	0.88
Han	Similarity	4	0.75
Proposed	CNN	9	0.962
		27	0.829

Han의 기법 같은 경우는 4개의 패밀리를 75%의 확률로 분류해 냈음을 볼 수 있는데 이 두 가지 분류 모델과 제안하는 모델을 비교하였을 때 제안하는 모델이 더 나은 분류 정확도를 보임을 알 수 있다. Nataraj의 모델과 비교하여 보면 Host-Rx 데이터 셋을 이용하여 분류한 실험에서는 6개의 패밀리를 95.1%의 정확도로 분류 한 것에 비해 제안하는 분류 기법은 9개의 패밀리를 96.2%의 정확도로 분류해냈음을 알 수 있다. 하지만 동일한 데이터 셋을 이용하여 많은 수의 패밀리를 분류하는 실험에서는 Nataraj의 모델은 531개의 패밀리를 72.8%로 분류해냈으나 제안하는 모델은 일정 수 이상의 샘플이 존재하는 27개의 패밀리에 대해서 82.9%의 분류 성능을 기록하였다.

기존의 연구들과 비교하였을 때 제안하는 모델이 가지는 장점은 악성코드를 분류하기 위한 특징점을 찾고 분류하는 과정이 단순하다는 점이다. Nataraj의 모델은 악성코드를 이미지화하고 Gabor filter를 이용하여 이미지 특징점을 추출하는 추가적인 과정이 필요한 반면 제안하는 모델은 악성코드 이미지를 분류기 학습의 입력으로 직접적으로 사용한다. 또한 Sandbox상에서 악성코드를 실행시켜 특징점을 찾거나 악성코드의 분석을 통해 사용되는 API를 특징점으로 사용하는 Rieck의 모델이나 Han의 모델과 비교하여도 제안하는 모델은 간단한 방법으로 악성코드의 패밀리를 분류해 낼 수 있는 기법임을 알 수 있다. 제안하는 모델의 또 다른 장점은 빠른 속도로 악성코드를 분류할 수 있다는 점이다. 특히 유사한 정확도를 기록한 Nataraj모델과 비교하면, 하나의 악성코드를 분류할 때 악성코드 이미지로부터 특징점을 추출하는 과정에서 평균 54ms가 소요되고, 하나의 샘플을 분류하기 위해서 학습된 모든 샘플과 비교해야하는 kNN 분류기의 특성으로 인해 하나의 샘플을 분류하는데 걸리는 전체 시간은 1.4s가 소요된다. 반면 제안하는 기법의 경우 1500개의 샘플을 분류하는데 5.98s가 걸려 하나의 샘플을 분류하는 평균 소요시간은 4ms를 기록하였다. 이는 기존 방법에 비해 약 350배 빠른 속도로 유사한 성능을 낼 수 있는 모델임을 알 수 있다.

VI. 결 론

본 논문에서는 심층 신경망을 적용한 기법 중 하나인 CNN를 이용하여 악성코드의 이미지만으로 악

성코드의 패밀리를 분류하는 새로운 기법을 제안하였다.

제안하는 모델의 악성코드 패밀리 분류 성능을 시험하기 위해 두 가지의 다른 데이터 셋을 이용하여 악성코드 패밀리를 분류한 결과 Microsoft 데이터 셋의 경우 9개의 패밀리를 96.2%의 분류 정확도로 구분해 냈고, VXHeavens 데이터 셋의 경우 27개의 패밀리를 82.9%의 정확도로 분류해 냈다. 두 데이터 셋의 분류 정확도 차이는 실험 결과를 분석해 본 결과 Microsoft 데이터 셋이 VXHeavens 데이터 셋에 비해 각각의 패밀리별 샘플의 수가 더 많고, 분류 대상 패밀리의 수 또한 VXHeavens 데이터 셋 보다 적어 더 높은 분류 정확도를 기록하였다. 제안하는 기법이 가지는 분류 정확도는 Nataraj의 분류 모델이나 Rieck의 모델, Han의 모델과 같은 기존 연구들과 비교하여 보아도 더 우수한 성능을 보임을 알 수 있다.

기존 연구에서도 악성코드를 이미지로 변환시켜 패밀리는 분류하는 기법이 존재하였으나 해당 연구에서는 악성코드 이미지를 분류기로 직접 학습시킬 수 없고, 특징점을 추출하는 과정이 추가적으로 필요하다는 단점이 존재하며 사용하는 악성코드 샘플의 수가 늘어날수록 패밀리 분류에 소요되는 시간이 길어진다는 단점이 존재하였다.

제안하는 분류 기법은 기존 연구들과는 달리 악성코드를 실행시키지 않고 악성코드 패밀리를 분류하는데 사용되는 특징점이 오직 악성코드 이미지만을 사용한다는 점과 CNN를 분류기로 사용하므로 추가적인 특징점 추출기가 필요하지 않다. 따라서 하나의 악성코드를 분류하는데 소요되는 시간은 약 4ms로 유사한 분류 정확도를 보인 Nataraj모델과 비교하였을 때 약 350배 빠르게 악성코드 패밀리를 분류할 수 있다는 장점을 가진다.

그러므로 제안하는 악성코드 패밀리 분류 모델은 하루에도 수십만 개의 악성코드가 생성되고 배포되는 상황에서 악성코드를 실행시키거나 코드를 분석하지 않고도 빠르게 악성코드 패밀리를 분류하여 적절한 방어기법을 제공할 수 있게 도움을 줄 수 있다.

하지만 실험결과에서 볼 수 있듯이 샘플데이터의 수가 충분히 확보되지 않으면 악성코드 패밀리 분류기가 충분히 학습되지 못해 정확도가 낮아질 수밖에 없으므로 이에 대한 보완책에 대한 연구가 필요하다. 또한 악성코드 패밀리 수가 많아지는 경우 CNN의 계층도 많아 져야 하므로 높은 하드웨어 성능이 요구되어 적은 샘플로도 악성코드들을 효과적으로 구분할 수 있도록 추가적인 연구가 필요하다.

References

〔1〕 A. Test, "Malware Statistics." https://www.av-test.org/en/statistics/malware//, 2015. 〔Online; accessed 22-September-2015〕.

〔2〕 K. S. Han, B. Kang, and E. G. Im, "Malware classification using instruction frequencies," in Proceedings of the 2011 ACM Symposium on Research in Applied Computation, RACS '11, (New York, NY, USA), pp. 298-300, ACM, 2011.

〔3〕 J. Kinable and O. Kostakis, "Malware classication based on call graph clustering," Journal in Computer Virology, vol. 7, no. 4, pp. 233-245, 2011.

〔4〕 M. Islam, R. Tian, L. Batten, and S. Versteeg, "Classication of malware based on string and function feature selection," in Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Second, pp. 9-17, July 2010.

〔5〕 R. Tian, L. Batten, and S. Versteeg, "Function length as a tool for malware classication," in Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on, pp. 69-76, Oct 2008.

〔6〕 M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in Recent Advances in Intrusion Detection (C. Kruegel, R. Lippmann, and A. Clark, eds.), vol. 4637 of Lecture Notes in Computer Science, pp. 178-197, Springer, 2007.

〔7〕 M. Zolkipli and A. Jantan, "An approach for malware behavior identification and classification," in Computer Research and Development (ICCRD), 2011 3rd International Conference on, vol. 1, pp.

191-194, March 2011.
〔8〕 R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Applications, vol. 36, no. 2, pp. 646-656, 2013.
〔9〕 H. Seo, J. Choi, and P. Chu, "A Study on Windows Malicious Code Classification System", Journal of the Korea Society for Simulation, vol. 18, no. 1, pp. 63-70, 2009.
〔10〕 R. Pascanu, J. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pp. 1916-1920, April 2015.
〔11〕 W. Jung, S. Kim, and S. Choi, "Poster: Deep learning for zero-day ash malware detection," 2015.
〔12〕 L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classication using binary texture analysis and dynamic analysis," in Proceedings of the 4th ACM Workshop on Security and Articial Intelligence, AISec '11, (New York, NY, USA), pp. 21-30, ACM, 2011.
〔13〕 A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classication with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097-1105, 2012.
〔14〕 Microsoft, "Microsoft malware classication challenge (big 2015)," 2015-02-03. 〔Online; accessed 2-July-2015〕
〔15〕 VX Heaven, "Vx heaven virus collection 2010-05-18." http://vxheaven.org/. 〔Online; accessed 18-May-2015〕.
〔16〕 virustotal, "VirusTotal." https://www.virustotal.com/. 〔Online; accessed 04-November-2015〕.
〔17〕 M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classication tasks," Information Processing & Management, vol. 45, no. 4,pp. 427-437, 2009.
〔18〕 K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 108-125, Springer, 2008.
〔19〕 B. Han, Y. Choi, and B. Bae, "Generating Malware DNA to Classify the Similar Malwares" Journal of the Korea Institute of Information Security and Cryptology, vol. 23, pp. 679-694, 2013.
〔20〕 L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classication," in Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, (New York,NY, USA), pp. 4:1-4:7, ACM, 2011.

〈저자소개〉

석 선 희 (Seonhee Seok) 학생회원
2011년 2월: 부산대학교 정보컴퓨터공학부 (공학사)
2014년 3월~현재: 부산대학교 전기전자컴퓨터공학과 석사과정
〈관심분야〉 사물인터넷, 머신러닝/딥러닝, 정보보호

김 호 원 (Howon Kim) 종신회원
1993년 2월: 경북대학교 전자공학과 (공학사)
1995년 2월: 포항공과대학교 전자전기공학과 (공학석사)
1999년 2월: 포항공과대학교 전자전기공학과 (공학박사)
1998년~2008년: 한국전자통신연구원(ETRI) 정보보호연구단 선임연구원/팀장
2008년~현재: 부산대학교 정보컴퓨터공학부 부교수
〈관심분야〉 사물인터넷, 정보보호/해킹 대응, 암호, 지능형 시스템/머신러닝