# 이미지와 메타데이터를 활용한 CNN 기반의 악성코드 패밀리 분류 기법

이송이, 문봉교, 김준태
동국대학교 컴퓨터공학과
sylee37@dgu.edu, bongkyo.moon@gmail.com, jkim@dongguk.edu

# Malware Classification Schemes Based on CNN Using Images and Metadata

Song Yi Lee, Bongkyo Moon, Juntae Kim
*Dept. of Computer Science Engineering, Dongguk University

요 약

본 논문에서는 딥러닝의 CNN(Convolution Neural Network) 학습을 통하여 악성코드를 실행시키지 않고서 악성코드 변종을 패밀리 그룹으로 분류하는 방법을 연구한다. 먼저 데이터 전처리를 통해 3가지의 서로 다른 방법으로 악성코드 이미지와 메타데이터를 생성하고 이를 CNN으로 학습시킨다. 첫째, 악성코드의 byte 파일을 8비트 gray-scale 이미지로 시각화하는 방법이다. 둘째, 악성코드 asm 파일의 opcode sequence 정보를 추출하고 이를 이미지로 변환하는 방법이다. 셋째, 악성코드 이미지와 메타데이터를 결합하여 분류에 적용하는 방법이다. 이미지 특징 추출을 위해서는 본고에서 제안한 CNN을 통한 학습 방식과 더불어 3개의 Pre-trained된 CNN 모델을 (InceptionV3, Densnet, Resnet-50) 사용하여 전이학습을 진행한다. 전이학습 시에는 마지막 분류 레이어층에서 본 논문에서 선택한 데이터셋에 대해서만 학습하도록 파인튜닝하였다. 결과적으로 가공된 악성코드 데이터를 적용하여 9개의 악성코드 패밀리로 분류하고 예측 정확도를 측정해 비교 분석한다.

## 1. 서론

악성코드 변종이란 기존의 악성코드를 변형하여 악성코드 탐지를 회피하기 위한 목적으로 출현되었으며 악성코드 변종 그룹을 악성코드 패밀리라 부른다. 이러한 악성코드 탐지 목적으로 기존의 시그니처 기반 방식은 변형되는 악성코드를 탐지하는 데 어렵다는 문제가 있다. 따라서 최근에는 변종 프로그램에 대응하기 위해 딥러닝을 이용한 패밀리 분류 기법에 대한 연구가 활발하게 이루어지고 있다.

## 2. 관련 연구

딥러닝을 이용한 악성코드 분류 기법으로는 악성코드를 실행시키지 않고 악성코드 속성 정보를 이용해 분석하는 정적 분석이 많이 활용되고 있다. Nataraj et al[1]의 연구에서는 악성코드 바이너리를 그레이 스케일 이미지로 시각화하는 방법을 제시하고 KNN 모델을 통해 패밀리를 분류했다.

Danish et al[2]와, Ahmet et al[3]의 연구에서는 VGG16, ResNet-50, GoogleNet 등의 Pre-trained CNN 모델을 Ensemble하여 악성코드를 분류한다. 그러나 이미 가공된 이미지 데이터셋을 활용했다는 점에서 데이터 전처리에 대한 논의가 부족하다.

Jeong et al[4]의 연구에서는 악성 파일의 opcode 시퀀스의 n-gram을 추출하여 특징 정보로 이용하였고 Simhash 인코딩, PCA, Linear Transform을 사용한 분류 방법을 제안한다. 이처럼 데이터 전처리 기법으로 이미지와 코드 정보를 활용한 연구들을 쉽게 접할 수 있다.

한편 새로운 전처리 방식을 사용하여 가공된 데이터를 EfficientNet, SENet, ResNet 등 여러 CNN 모델에 적용한 사례도 있다. Gessert et al[5]의 연구에서는 피부 병변 분류를 위해 피부 병변 이미지와 함께 환자 정보가 담긴 메타데이터를 결합하여 사용했다는 점이 흥미롭다. 이는 이미지 데이터만 적용했을 때보다 더 높은 분류 정확도를 보였다.

본 논문에서는 악성코드 분류를 위해 3가지의 서

* 이 논문은 2017년 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보 컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (NRF-2017M3C4A708279).

로 다른 데이터 전처리 기법을 이용한다. 또한, 다양한 CNN 모델을 통한 학습을 진행하여 악성코드 분류 성능 차이를 비교하고 평가한다.

## 3. 제안 모델 및 방법

본 논문에서 활용한 악성 파일의 데이터 전처리 기법은 다음과 같다. 첫째, 악성코드 파일을 별도의 전처리 과정 없이 이미지로 변환하는 방법이다. 둘째, 악성코드 파일의 opcode sequence를 이미지로 시각화하는 방법이다. 이에 더해 본고에서는 세 번째 방법으로 악성코드 이미지와 메타데이터를 결합하는 방식을 활용한다. 메타데이터는 악성 파일에 대한 정보를 설명하고 있으므로 악성 파일을 분석하는 데 유용하게 작용 될 수 있을 것으로 생각된다. 이후 전처리 과정을 거친 데이터를 서로 다른 CNN 모델에 적용해 이미지 특징을 추출한다.

### 3.1 Dataset

데이터셋은 Microsoft Malware Classification Challenge (BIG 2015)를 사용하였다. Microsoft 데이터셋은 악성코드 변종 파일을 16진수로 변환한 10,868개의 BYTES 파일과 어셈블리어로 변환한 10,868개의 ASM 파일로 구성되어 있다. <그림1>과 <그림2>를 통해 파일의 형태를 확인할 수 있다. 악성코드 파일은 9개의 패밀리 중 하나에 속한다. 패밀리는 Ramnit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY, Gatak으로 총 9가지로 분류된다.

```
00401000 56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08
00401010 BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC
00401020 C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC
00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44
00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6
00401050 5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC CC
00401060 8B 44 24 08 8A 08 8B 54 24 04 88 0A C3 CC CC CC
00401070 8B 44 24 04 8D 50 01 8A 08 40 84 C9 75 F9 2B C2
00401080 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
```

<그림 1> Bytes file sample (Hex code)

```
.text:00401051                          ; ---------------------------------------
.text:00401054 CC CC CC CC CC CC CC CC CC CC CC CC      align 10h
.text:00401060 8B 44 24 08              mov     eax, [esp+8]
.text:00401064 8A 08                    mov     cl, [eax]
.text:00401066 8B 54 24 04              mov     edx, [esp+4]
.text:0040106A 88 0A                    mov     [edx], cl
.text:0040106C C3                     retn
.text:0040106C                          ; ---------------------------------------
.text:0040106D CC CC CC                 align 10h
.text:00401070 8B 44 24 04              mov     eax, [esp+4]
.text:00401074 8D 50 01                 lea     edx, [eax+1]
.text:00401077
```

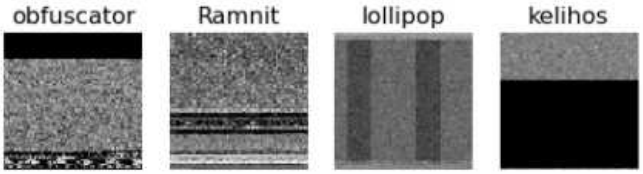
<그림 2> Asm file sample (assembled code)

* 이 논문은 2021년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2021R1A2C2008414).

### 3.2. Data pre-processing

#### 3.2.1. 악성코드 파일의 이미지 변환

악성코드 이미지 변환기를 이용해 byte 파일을 이미지 파일로 변환하는 작업을 수행한다.[1] 악성코드 바이너리 파일을 8-bit 정수의 벡터로 읽고 이것을 2차원 배열로 변환 후 <그림3>와 같은 gray-scale 이미지로 시각화한다. CNN 모델은 고정된 크기의 입력에서만 동작하기 때문에 이미지를 고정된 크기로 변환한다. 한 변의 길이를 [√S]라고 했을 때 [√S] * [√S] 형태의 사각형 이미지로 생성하며 바이트의 크기가 크면 축소하고 작을 경우 0값의 padding을 추가해 확대한다. 256 * 256 크기의 정사각형 이미지로 고정하며 변환 단계를 마친다.




<그림 3> 악성코드 Gray-scale 예시

#### 3.2.2. opcode sequence 추출과 이미지 변환

실행 파일의 opcode는 어떤 명령 함수를 통해 행위를 수행하는지 알 수 있으므로 악성코드의 공격 행위를 파악할 수 있다. 본 논문에서는 제공된 데이터셋 중 ASM 파일을 사용하여 opcode sequence를 이미지로 변환하는 작업을 수행한다.[4]

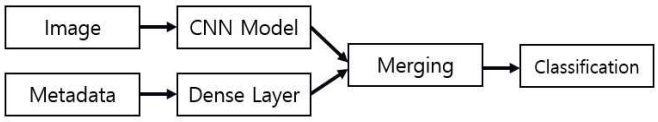
첫째, ASM 파일로부터 .text section의 opcode sequence를 추출한다. 둘째, 추출한 opcode sequence 정보를 SimHash (SHA-256) 알고리즘을 통해 인코딩하여 CNN에 적용하기 적합한 고정된 길이로 반환한다. 실행 파일의 opcode 정보는 해시 테이블의 고정된 크기로 매핑되기 때문에, 파일별 opcode 수가 다르더라도 고정된 크기로 반환된다. 마지막으로, 반환된 다이제스트를 바이너리 형태로 변환하며 0 이하의 값은 0, 0 초과 값은 1로 반환하고 바이너리 형태의 opcode sequence를 <그림 4>와 같은 사각형의 gray-scale 이미지로 변환한다.

<그림 4> Opcode gray-scale 예시

### 3.2.3. 악성코드 이미지와 메타데이터의 결합

본고에서 제안하는 실험은 악성코드 이미지와 메타데이터를 결합한 통합 데이터를 사용하는 방법이다. 이를 위해 본 논문에서는 BYTES 파일을 통해 변환된 악성코드 이미지와 ASM 파일에서 추출한 메타데이터를 함께 활용한다. 또한, 실험을 위해 신경망 훈련 간소화에 도움을 주는 fastai 라이브러리를 사용하며 다음 작업을 수행한다.

첫째, ASM 파일에서 segments, opcode, file size 정보를 추출하고 추출된 각 데이터의 빈도수를 카운팅한 결과물을 CSV 파일로 저장한다. 이를 통해 총 48개의 특징을 가진 악성 파일의 메타데이터가 생성된다. 둘째, 각 이미지에 해당하는 메타데이터를 매핑시켜 두 개의 데이터를 통합한다. 셋째, 이미지 데이터 학습을 위한 CNN 모델과 메타데이터 학습을 위한 Tabular 모델을 각각 생성한다. 넷째, 통합 데이터 학습을 위해 2개의 입력을 받아 1개의 출력을 보이는 CNN 통합 모델을 생성한다. 통합 모델은 마지막 레이어층인 분류기를 사용하여 두 모델의 출력을 연결한다. Dense 계층은 입력 뉴런 수에 상관없이 출력 뉴런 수를 자유롭게 설정할 수 있어 출력층으로 사용된다. 위 과정을 간략화하면 <그림 5>와 같다.




<그림 5> 데이터 통합 모델 구조

### 3.3. Transfer Learning & Fine-Tuning

본고에서는 Pre-trained CNN 모델로 InceptionV3, ResNet, DenseNet를 사용하여 전이학습을 진행한다. 이러한 네트워크는 [2],[3]을 비롯해 다양한 악성코드 분류에 탁월한 성능을 제공하는 것으로 나타났다. 본고에서는 각 CNN 모델을 이미지 특징 추출 메커니즘으로써 이용하기 위해 기존 계층의 가중치를 동결(freeze)하고 마지막 분류 계층을 추가 및 미세 조정하여 악성코드 데이터에 대해 새롭게 학습시킨다. Fine-Tuning된 CNN 구조는 <그림 6>과 같다. 기존 모델 위에 추가된 레이어는 차원 축소, 오버피팅 방지, 분류에 사용된다. 결과적으로 기존 CNN 모델 내 하위의 특징 추출 레이어들은 학습되지 않고 추가된 마지막 분류 레이어만을 통해 학습된다.

### 3.4. Convolutional Neural Networks

본고에서는 다양한 아키텍처 간의 학습 정확도 측정을 위해 <그림 7>의 구조를 갖는 CNN 모델을 제안하고 학습을 진행한다.

| **Feature Extractor** |
|---|
| CNN based |
| **(+) Add layers** |
| GlobalMaxPooling2D (Flatten the output layer) |
| Dense (128, units, Relu activation) |
| Dropout (0.2 rate to reduce overfitting) |
| Dense (softmax layer for classification) |

<그림 6> Fine-tuning된 Pre-trained CNN 모델 구조

| **ConvolutionNeural Network** |
|---|
| Input layer (62, 62, 30) |
| Convolution layer (62, 62, 30) |
| GlobalMaxPooling2D (31, 31, 30) |
| Convolution layer (29, 29, 15) |
| GlobalMaxPooling2D (14, 14, 15) |
| Dropout (0.2 rate 14, 14, 15) |
| Flatten (2,940) |
| Dense (128 units, Relu activation) |
| Dropout (0.5 rate, 128) |
| Dense (50 units, Relu activation) |
| Output layer (softmax units) |

<그림 7> 제안하는 CNN 모델 구조

## 4. 실험 및 분석

### 4.1. 실험 환경

실험 환경은 Windows 10(64bit) 운영체제에서 Tensorflow Backend Keras 중심으로 실험하였다. 상세 실험 환경은 <표 1>에 기술되어있다.

| 분류 | 내용 |
|---|---|
| OS | Windows 10 |
| CPU | Inter Core i5-10210U 1.60GHz |
| RAM | 16GB |
| Tensorflow | 2.2 |
| Pytorch | 1.4 |
| Keras | 2.3.1 |

<표 1> 실험 환경

### 4.2. 결과 및 분석

본고에서는 다양한 데이터 전처리 기법을 활용하고

서로 다른 CNN 모델에 적용하여 악성코드 패밀리 유형 9개 중 하나로 분류하는 실험을 진행한다. 학습을 위해 악성 파일 중 8,695개를 training set로 이용하고 2,173개는 testing set로 사용하였다. 이후 10-fold cross validation 기법을 적용해 데이터셋을 10개로 나누어 성능 측정을 반복한다. 분류 정확도 결과는 <표2>와 같다.

먼저, 첫 번째 전처리 방식을 거친 학습 결과는 제안한 CNN 모델에서 95.6%로 가장 높은 성능을 보인다. 전이학습의 결과는 ResNet-50 (89%), InceptionV3(89%), DenNet(74%)로 CNN 모델보다 낮은 정확도를 보였다. 두 번째 전처리 방식을 거친 학습 결과 또한 제안한 CNN 모델에서 92.4%로 가장 높은 결과를 보였으며, ResNet-50(85.4%), DensNet(82%), InceptionV3(79.3%) 순의 정확도를 나타냈다. 또한, 전체 성능 비교 시에는 첫 번째 전처리 방식의 학습이 가장 높은 정확도를 보였다.

반면 본고에서 제안한 세 번째 전처리 방식을 거친 통합 모델의 학습은 완벽한 결과 값을 나타내지 못했다. 그 이유로는 먼저, GPU 메모리 문제로 실험 환경 업그레이드가 필요했다. 초기 실험에서 사용한 GPU는 2GB로 전처리 방식 1과 2를 통해 다양한 모델을 학습시키는 데 문제가 없었으나 전처리 방식 3을 이용한 통합 모델 학습 시에는 11GB의 GPU를 사용함으로써 메모리 오류 없이 원활하게 학습을 진행할 수 있었다. 두 번째 문제로는 학습 진행 시 발생한 오버피팅의 문제다. 본고에서 사용한 메타데이터는 48개의 특징을 가지고 있으나 문제 해결을 위해 특징 축소가 필요해 보인다.

| Model | Method 1 | Method 2 |
|---|---|---|
| CNN | 95.6% | 92.4% |
| InceptionV3 | 84.2% | 79.3% |
| ResNet-50 | 89% | 85.4% |
| DensNet | 74% | 82% |

<표 2> 실험 결과 비교

## 5. 결론

본 논문에서는 CNN을 적용하여 악성코드 패밀리를 분류하는 방법으로 3가지의 데이터 전처리 기법을 활용했다. 본고에서 새롭게 실험한 전처리 방식은 악성 파일의 이미지와 메타데이터를 결합해 통합 데이터를 생성하는 것으로 다양한 특징을 가진 통합 데이터가 학습 데이터로써 활용 가능함을 보인다.

또한, 본고에서 제안한 간단한 CNN 모델만으로 높은 정확도의 분류가 가능함을 알 수 있다. 이는 다수의 계층으로 이루어진 Pre-trained 모델의 전이학습과 비교했을 때 더 높은 정확도 결과를 보였으며 학습 시간 또한 10배 이상 단축되었다.

그러나 통합 모델의 학습 진행 시에는 높은 GPU 메모리가 필요했으며 과적합되는 문제가 발생하였다. 이를 해결하기 위해서는 고차원의 데이터 중 주요 특징만을 선정해 데이터 손실은 최소화하며 동시에 파라미터를 최적화할 수 있도록 추가적인 연구가 필요하다.

## 6. Future Work

본고에서는 통합 데이터 학습을 위한 CNN 통합 모델의 학습 결과를 기재하지 못하였으나 추후 연구로써 PCA를 활용해 데이터의 차원 수를 축소하는 방식으로 과적합 문제를 방지할 수 있을 것으로 생각한다. 이를 통해 변형된 데이터를 통합 모델에 적용해 학습시키고 결과물을 학술대회에서 발표할 예정이다.

## 참고문헌

[1] Nataraj L, Karthikeyan S, Jacob G, Manjunath B. S., “Malware images: visualization and automatic classification”, In proc. of the 8th ACM international symposium on visualization for cyber security 2011.

[2] Danish Vasanab, Mamoun Alazabe, Sobia Wassancd, Babak Safaei, Qin Zheng, “Image-Based malware classification using ensemble of CNN architectures (IMCEC)”, ELSEVIER, 2020.

[3] Ahmet Selman Bozkir, Ahmet Ogulcan Cankaya, Murat Aydos, “Utilization and Comparision of Convolutional Neural Networks in Malware Recognition”, IEEE, 2019.

[4] Young-Man Kwon, Jae-Ju An, Myung-Jae Lim, Seongsoo Cho, Won-Mo Gal, “Malware Classification Using Simhash Encoding and PCA (MCSP)” MDPI, 1-12, 2020.

[5] Gessert, N, Schlaefer, A, Nielsen, M., Shaikh, M, Werner, R, “Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data”, Elsevier B.V, 2020.