# 머신러닝을 활용한 악성코드 분류 방법

김수빈, 송재승*
세종대학교, *세종대학교

18011630@sju.ac.kr, *jssong@sejong.ac.kr

# A Study on the Malicious code classification method Using Machine Learning

Kim Su Bin, Song Jae Seung*
Sejong Univ., *Sejong Univ.

요 약

최근 인공지능, 그 중에서도 머신러닝의 중요성과 필요성이 강조되고 있다. 실제로 지능형 CCTV, AI 보안 시스템 등 인공지능을 활용하여 다양한 문제점들을 해결하거나 편리함을 제공하고 있으며 이를 통해서 인간들이 수동으로 일일이 대응해야 했던 것들을 인공지능이 해결할 수 있게 되었다. 그 중에서도 정보보안 분야는 악성코드의 발생 빈도와 처리 능력이 인간의 한계를 뛰어넘는 분야이기 때문에 특히 인공지능의 적용이 요구되는 분야 중 하나이다. 그래서 본 논문에서는 인공지능 및 머신러닝에 대해서 알아보고, 이러한 기술을 활용하여 악성코드를 분류하고 분석하는 방법에 대해서 제안하고자 한다.

## Ⅰ. 서 론

악성코드의 급속한 확산으로 매년 출시되는 새로운 시그니처 수가 급격히 증가하고 있다. 기존 접근방식인 정적분석 및 동적분석은 코드 난독화로 인한 어려움을 겪거나, 시간과 자원을 많이 소모하여 확장성 문제가 발생하거나, 모든 동작을 관찰할 수 없다는 등의 한계를 가진다. 본 논문에서는 이러한 문제를 해결하기 위하여 코드를 분석하거나, 조건을 설정하고 시간과 자원을 소모하며 직접 실행할 필요가 없도록 악성코드를 이미지화 하여 패턴을 학습하고, 학습 결과에 따라 분류할 수 있는 알고리즘을 제안한다.

본 논문은 머신러닝을 활용한 악성코드 분류 기법을 제안하기 위하여 가장 적합한 파라미터의 튜닝, 이미지 처리와 이미지 학습 및 분류기의 구조에 역점을 두어 연구를 수행하였다. CNN 기반 Inception 모델의 Fine tune 방식으로 이미지를 학습하고, 학습 정확도를 높이기 위하여 개발한 CRIS Ensemble 알고리즘으로 이미지 처리를 진행한다. 이후 KISA의 악성코드 이미지셋을 이용하여 성능을 확인하는 절차로 연구를 실시하였다.

## Ⅱ. 이미지 분류 알고리즘

머신러닝을 활용한 악성코드 분류를 위한 학습기와 분류기를 개발하는 것은 이미지화 단계와 텐서플로우와 같은 라이브러리를 활용한 이미지 학습/분류단계로 구분할 수 있다.

특히 본 연구는 악성코드를 학습하고 분류하는 머신러닝 시스템의 가장 적합한 알고리즘을 연구하는 것이 목적이다. 이 목적을 달성하기 위해서 다양한 연구를 통해 다음과 같은 머신러닝 알고리즘의 라이브러리와 프로세싱 모델을 결정하였고 두 번째로는 알고리즘의 파라미터를 결정하고 이를 튜닝하면서 알고리즘의 적합도를 실험하였다.

### 3.1 머신러닝 알고리즘 선택

머신러닝 악성코드 분류기를 만들기 위해서 머신러닝 알고리즘의 대표주자 격인 구글의 텐서플로우를 선택하여 알고리즘을 개발하였다. 그러나 악성코드 이미지의 복잡성 그리고 포함된 데이터의 불확실성에 따라 그 정확도가 기대에 못 미치게 되었다. 과제의 성공적인 수행을 위해 텐서플로우 라이브러리에서 이미지분류 분야의 특화된 모델을 선정하고 그것을 이용한 시스템을 개발하고 정확도를 측정하기 위해 노력하였다. 그 결과로 텐서플로우 뿐 아니라 여러 가지 머신러닝 오픈소스 라이브러리 중에서도 이미지 분류에 가장 성능이 좋고 특화된 인셉션(Inception)을 선택하였고 이를 통한 악성코드 이미지 학습 및 분류기를 개발하고 그 알고리즘의 적합성을 점검하였다.

인셉션은 구글의 컴퓨터 비전 기술이다. 구글은 지난 2014년 글로벌 이미지인식 경진대회 GoogLeNet에서 처음 우승을 했는데, 이것이 인셉션 V1이다. 당시 GoogLeNet은 89.6%의 인식률을 기록했다. 현재는 V4까지 나와 있다. 인셉션 모델은 학습 방식에 따라 Scratch 방식과 Fine tune 방식으로 분류할 수 있다 [1].

- Scratch 방식

  스크래치란 말 그대로 분류하고 싶은 이미지를 텐서플로우가 인식 가능한 이미지로 변환한 후, 학습과정을 거쳐 신경망으로 저장한다. 이후에는 분류과정을 거치게 되는데 학습결과에 다시 한번 더 소정의 학습 과정을 거치는 Fine tune방식에 비해서는 정확도가 떨어진다.

- Fine tune 방식

  이 방식의 정확한 의미는 Fine-Tune a Pre-Trained Model 이라는 표기에서 찾을 수 있다. 기존 학습모델에 가중치 조정을 시키거나 자신만의 새로운 학습모델을 만들 수 있도록 전 처리된 데이터 셋을 만든다. 원하는 이미지를 분류할 때는 가능하지만 학습할 때에는 이미지를 바로 인식하여 학습하지 못한다. 그래서 텐서플로우가 인식할 수 있도록 변환하는데 이미지와 이미지의 레이블을 구조화하여 저장한다. 본 연구에서는 높은 정확도를 위하여 Fine tune 방식을 사용하였다 [2].

## Ⅲ. 실험 수행 및 결과

본 절에서는 실험 수행 과정과 결과에 대하여 소개한다. 개발 및 실험은 기본적으로 Windows에서 수행되었으며, Tensorflow를 구현하기 위하여 Python3을 사용하였다.

### 3.1 분류 이미지 변환

학습할 악성코드 이미지를 바이너리 포맷으로 변환하기 위한 imgReader.py 소스코드는 [Table 1]과 같다. convertImgs 함수는 JPG 이미지를 BIN 파일로 변환하는 역할을 수행하며, 함수 호출 시 지정된 디렉토리에 BIN 파일이 생성된다.

[Table 1] imgReader.py

```
def convertImgs(sess, dataName):
   files = getFiles(dataName)
   with open( FLAGS.train_dir + '/' + dataName + '.bin'
, 'wb') as file:
      count = 0
      for i in range(0, len(files)):
         resImg = resizeImg(files[i][1])
         try:
            image = sess.run(resImg)
         except Exception as ex:
            print (ex.message)
            continue
         k = int(files[i][0])
         file.write(chr(k).encode())
         file.write(image.data)
         count += 1
```

3.2 변환된 이미지 학습

앞절에서 생성된 바이너리 파일들을 활용하여 학습을 수행한다. 해당 과정에서 CNN 알고리즘이 사용되며, 학습이 완료되면 result 디렉토리에 result.dat 파일이 생성된다. Train.py 소스코드는 [Table 2]와 같다.

[Table 2] Train.py

```
def train():
   fd = tf.placeholder(tf.float32)
   images, labels = td.batch_data('train', FLAGS.batch
_size)
   logits = cnn.inference(images, fd)
   loss = cnn.loss(logits, labels)
   train = cnn.train(loss)
   with tf.Session() as sess:
      saver = tf.train.Saver()
      if tf.gfile.Exists(FLAGS.result_dir +
'/result.dat'):
         saver.restore(sess, FLAGS.result_dir +
'/result.dat')
      else:
         init = tf.initialize_all_variables()
         sess.run(init)
      coord = tf.train.Coordinator()
      threads = tf.train.start_queue_runners(sess=sess
, coord=coord)
      for step in range(FLAGS.max_steps):
         sess.run(train, feed_dict={fd: 0.7})
         print(step, sess.run(loss, feed_dict={fd:
1.0}))
         if step % 100 == 0 or (step + 1) == FLAGS.
max_steps:
            saver.save(sess, FLAGS.result_dir +
'/result.dat')
```

3.3 실험 결과

본 절에서는 실험한 결과와, 결과에 대한 고찰에 대하여 다룬다. 정확도를 기준으로 메소드의 적절성을 판단하고, 정확도 향상을 위한 방안을 소개한다.

실험 결과 Scratch 방식에 비해 Fine tune 방식이 보다 비교적 높은 정확도를 보였으며, Batch 사이즈와 Steps가 커질수록 정확도가 낮아지는 양상을 보였다. 위 세가지 요소를 조정하였을 때 분류 정확도는 80%대에 머물렀다. 악성코드 이미지 분류의 정확도를 90% 이상 기록하기 위하여 학습 전 데이터 처리 과정을 추가하였다.

기존 Ensemble 방식은 이미지 처리를 앙상블 할 때 고정 시작점을 갖는 방식이었으나 본 연구에서 사용한 CRIS Ensemble 방식을 방식은 그 기준점을 변경한다. 이미지를 Crop, Rotate, Inverse, Resize 하여 새로운 앙상블 방식을 적용하였다. 적용 결과는 [Table 3]과 같다.

[Table 3] 실행 결과

| 구분 | 원본 | 크로핑 이미지 앙상블 |
|---|---|---|
| Method | Fine tuen | Fine tune |
| Validation | 600 | 600 |
| Steps | 1000 / 500 | 1000 / 500 |
| Batch size | 64 | 64 |
| Learning rate | 0.01 / 0.001 | 0.01 / 0.001 |
| Weight decay | 0.00004 | 0.00004 |
| Model | Inception V1 | Inception V1 |
| Result | 82% | 97% |

## Ⅳ. 결 론

계속해서 심해지는 악성코드 공격, 이에 대한 가장 적극적이며 각광받는 신기술 솔루션으로 머신러닝이 언급된 것은 그리 오래전의 일이 아니다. 국외는 물론 국내에서도 이 분야의 연구가 활발히 진행되고 있다.

머신러닝을 이용한 악성코드의 분석에 관한 연구는 크게 2가지의 핵심적 고민이 필요하다. 첫째는 이미지화의 방식이다. 악성코드 이미지화 연구부분을 보면, 고유의 시그니쳐를 시각화 할 수 있는 키는 악성코드의 이미지화시 사용하는 Width 값의 결정이다. 이 값을 유연하게 변화시켜가면서 가장 적절한 이미지화를 할 수 있다면 악성코드 분류의 정확도가 상승한다. 또한 악성코드 PE파일 5개 섹션의 정확한 선택과 악성코드 바이트 코드 중에서 disassemble 단계에서 나타난 부분을 어떻게 처리할지도 연구의 중요한 요소이다.

둘째는 CNN의 구출분야이다. Tensorflow 권고에 의하면 정확한 분류를 위한 Factor로 100만 번의 콘볼루션 연산을 요구하고 있다. 그러나 이렇게 할 경우 학습기와 시험기의 작동에 너무 많은 소요시간이 요구되어 시스템의 성능에 큰 영향을 주게 된다. 실험결과의 많은 부분을 고려할 때 콘볼루션 연산과 분류결과의 정확성은 상충되는 것이라는 판단을 할 수 있다. 이 문제를 해결하기 위해 텐서플로우 SLIM 라이브러리의 인셉션 모델을 적극 도입하였고, 이를 통해 80% 이상의 정확도를 기록하였다. 결론적으로 인셉션 라이브러리를 활용하였고, 학습방식은 Fine tune, 무엇보다 가장 중요한 이미지처리 알고리즘으로는 자체개발한 CRIS Ensemble을 적용하여 데이터 셋 689개로 97%의 정확도를 기록할 수 있었다.

이러한 결과를 바탕으로, 머신러닝을 활용한 악성 코드 분류 시스템은 그 결과의 정확도와 유용성 면에서 요구조건을 만족시킬 수 있는 효과적인 방안이 될 수 있음을 확인할 수 있었다.

## ACKNOWLEDGMENT

본 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2019-0-00426, IoT 기반 이식-침습형 고위험 의료장치를 위한 능동형 킬 스위치 및 바이오 마커 활용 방어 시스템 개발)

## 참 고 문 헌

[1] S. I. V. V. A. A. C Szegedy, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] A. C. K. B. Aidan Boyd, “Deep Learning-Based Feature Extraction in Iris Recognition: Use Existing Models, Fine-tune or Train From Scratch?,” *IEEE International Conference on Biometrics*, 2019.