2025 년도

학사학위논문

머신러닝 기반 악성코드 탐지 모델의 성능 비교

Performance comparison of machine learning-based malware detection models

2025 년 05 월 24 일

순천향대학교 공과대학

컴퓨터공학과

강범석

머신러닝 기반 악성코드 탐지 모델의 성능 비교

Performance comparison of machine learning-based malware detection models

지도교수 천인국

이 논문을 공학사학위 논문으로 제출함

2025 년 05 월 24 일

순천향대학교 공과대학

컴퓨터 공학과

강범석

강범석의 공학사학위논문을 인준함

2025 년 05 월 24 일

지 도 교 수 천 인 국 인

심 사 위 원 조 용 원 인

순천향대학교 공과대학

컴퓨터공학과

초 록

21 세기 정보화 사회의 발전과 함께 악성코드(Malware)는 점점 더 다양하고 정교한 형태로 진화하고 있다. 이에 따라 기존 시그니처 기반 정적 탐지 기법으로는 신종 및 변종 악성코드를 효과적으로 대응하는 데 한계가 존재한다.

본 논문에서는 머신러닝 및 딥러닝 기반 모델을 활용하여 악성코드 탐지 성능을 향상시키는 방안을 제안한다. 실험에는 Malimg 이미지 데이터셋을 활용하였으며, Histogram of Oriented Gradients (HOG) 특징을 추출하여 Random Forest, SVM, One-Class SVM 모델을 학습하였다. 또한 VGG 스타일의 심층 신경망(CNN)을 적용하여 이미지 기반 학습도 진행하였다.

각 모델에 대한 성능 평가 결과, VGG 이 약 98%의 정확도로 가장 우수한 결과를 보였으며, Random Forest 와 SVM 은 각각 약 91%, 89%의 정확도를 기록하였으며, One-Class SVM 은 정상 데이터로 설정한 패밀리를 기반으로 이상 탐지에 유효한 결과를 도출하였다. 본 연구를 통해 머신러닝 및 딥러닝 기반 탐지 모델이 정적 탐지 방식의 한계를 보완하고 악성코드 탐지 성능을 높이는 데 기여할 수 있음을 확인하였다.

키워드: 악성코드(Malware), 머신러닝(Machine Learning), 딥러닝(Deep Learning), CNN (Convolutional Neural Network)

Abstract

With the advancement of the information society in the 21st century, malware has evolved into increasingly diverse and sophisticated forms. As a result, traditional signature-based static detection methods face limitations in effectively responding to new and variant malware.

This study proposes an approach to enhance malware detection performance by employing machine learning and deep learning models. The experiments utilize the Malimg image dataset, where Histogram of Oriented Gradients (HOG) features were extracted to train Random Forest, SVM, and One-Class SVM models. Additionally, a deep convolutional neural network (CNN) based on the VGG architecture was applied for image-based learning.

According to the performance evaluation, the VGG model achieved the highest accuracy of approximately 98%, while Random Forest and SVM recorded around 91% and 89% accuracy, respectively. The One-Class SVM also demonstrated effectiveness in anomaly detection using selected malware families as normal data. The results confirm that machine learning and deep learning-based detection models can complement the limitations of static detection methods and significantly improve malware detection performance.

Keyword: Malware, Machine Learning, Deep Learning, Convolutional Neural Network

차 례

그림 차례

표 차례

제 1 장 서론

21 세기 정보화 사회의 발전과 함께 사이버 공간을 통한 다양한 위협이 급격히 증가하고 있다. 특히, 악성코드(Malware)는 점점 더 지능적이고 다양한 형태로 진화하고 있으며, 이에 따른 피해 규모 역시 지속적으로 확대되고 있다.

기존의 악성코드 탐지 기법은 주로 시그니처(Signature) 기반 정적 분석 방법에 의존하고 있다. 이러한 방법은 알려진 악성코드의 패턴을 데이터베이스화 하여 탐지하는 방식으로, 이미 알려진 위협에 대해서는 높은 정확도를 보인다. 그러나 새로운 형태의 악성코드나 기존 악성코드의 변종에 대해서는 탐지율이 급격히 저하되는 한계를 가지고 있다. 특히, 악성코드가 난독화, 암호화 기술을 활용하여 기존 패턴을 회피할 경우, 시그니처 기반 탐지 기법은 효과적으로 대응하지 못한다.

이러한 한계를 극복하기 위해 본 논문에서는 머신러닝 및 딥러닝 기반 악성코드 탐지 모델을 설계하고, 다양한 모델의 성능을 비교 분석하고자 한다. 실험에는 Malimg 이미지 데이터셋을 활용하였으며, 머신러닝 기법으로 Random Forest, Support Vector Machine (SVM), One-Class SVM 을 딥러닝 기법으로는 Visual Geometry Group (VGG) 스타일의 심층 신경망(CNN)을 적용하였다. 머신러닝 모델에는 이미지 데이터에 Histogram of Oriented Gradients (HOG) 특징 추출 기법을 적용하여 사용하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 악성코드와 머신러닝, 딥러닝 관련 이론적 배경을 설명한다. 3 장에서는 제안한 악성코드 탐지 기법과 실험 방법을 기술하고, 4 장에서는 실험 결과를 분석한다. 마지막으로 5 장에서는 본 연구의 결론을 제시한다.

제 2 장 관련 연구

2.1 악성코드

악성코드(Malware)는 컴퓨터 시스템을 손상시키거나, 사용자 정보를 탈취하거나, 시스템 자원을 악용하기 위해 제작된 악성 프로그램을 의미한다. 악성코드는 다양한 형태로 존재하며, 대표적으로 바이러스, 웜, 트로이 목마, 랜섬웨어 등이 있다. 바이러스는 실행 파일에 자신을 삽입하여 파일을 감염시키는 형태로 동작하며, 웜은 스스로 복제하여 네트워크를 통해 확산된다. 트로이 목마는 정상 프로그램으로 위장하여 시스템에 침투한 후 악성 행위를 수행한다. 랜섬웨어는 사용자의 파일을 암호화한 후 금전적인 대가를 요구하는 방식으로 피해를 유발한다.

표 1은 악성코드들의 종류와 특징을 정리한 표이다.

종류	**특징**
바이러스(Virus)	실행 파일에 자신을 삽입하여 다른 파일을 감염시키는 악성코드
웜(Worm)	네트워크를 통해 스스로 복제하고 확산하는 악성코드
트로이 목마(Trojan Horse)	정상 프로그램으로 위장하여 설치된 후 악성 행위를 수행하는 악성코드
랜섬웨어(Ransomware)	사용자의 파일을 암호화하고 금전적 대가를 요구하는 악성코드

[표 1] 악성코드의 종류와 특징

2.2 머신러닝

머신러닝(Machine Learning)은 명시적으로 프로그래밍 되지 않은 상태에서 컴퓨터가 주어진 데이터로부터 패턴을 학습하고, 이를 기반으로 예측하거나 분류하는 기술을 의미한다[1]. 전통적인 프로그래밍 방식에서는 개발자가 모든 규칙을 직접 정의해야 하지만, 머신러닝은 대규모 데이터를 이용하여 스스로 규칙을 찾아내는 것이 특징이다.

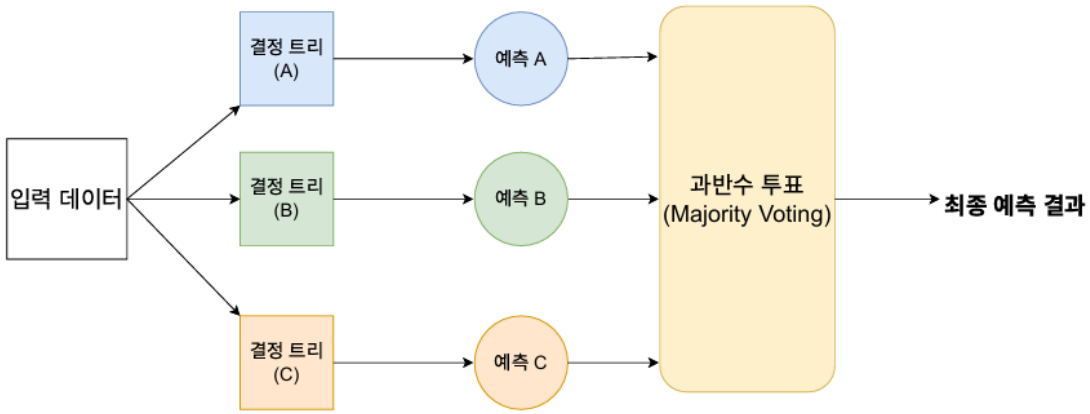
악성코드 탐지 분야에서는 머신러닝을 통해 신종 악성코드나 변종 악성코드에 대해 보다 유연하고 효과적으로 대응할 수 있다. 기존 시그니처 기반 탐지 방식은 알려진 악성코드만 탐지할 수 있는 한계가 있지만[2], 머신러닝 기법은 데이터 기반으로 일반화된 패턴을 학습함으로 써 알려지지 않은 악성코드에 대해서도 탐지가 가능하다.

본 연구에서는 악성코드 이미지 데이터에 HOG(Histogram of Oriented Gradients) 특징을 추출하여 머신러닝 모델의 입력 데이터로 사용하고, Random Forest, SVM(Support Vector Machine), One-Class SVM 알고리즘을 적용하여 악성코드를 분류 및 탐지하는 실험을 진행하였다.

2.2.1 Random Forest

Random Forest 는 여러 개의 결정 트리(Decision Tree)를 조합하여 예측을 수행하는 학습 방법이다[3]. 이 알고리즘은 데이터의 일부 샘플과 특징(feature)을 무작위로 선택하여 각각의 트리를 독립적으로 학습시키는 방식으로 과적합(overfitting)을 방지하고 모델의 일반화 성능을 향상시킨다.

학습이 완료된 후, 테스트 데이터에 대해 각 트리는 독립적으로 예측을 수행하며, 최종 결과는 모든 트리의 예측 결과를 다수결(Voting) 방식으로 통합하여 결정된다[4]. 이로 인해 Random Forest 는 높은 분류 정확도와 견고한 성능을 보이며, 복잡한 데이터에서도 안정적인 결과를 제공할 수 있다.

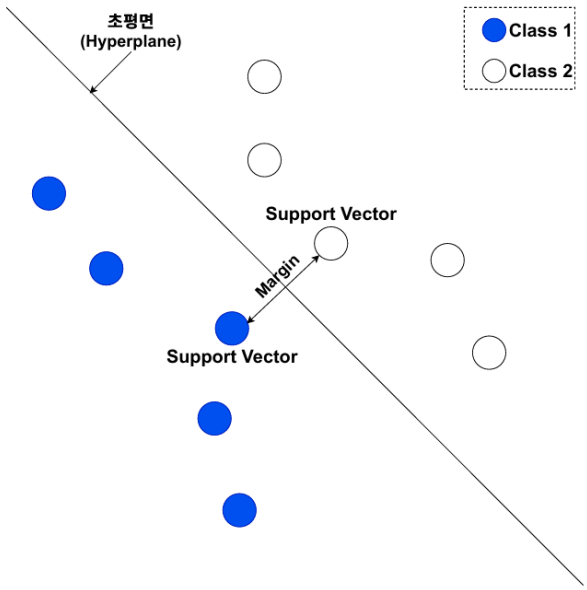


[그림 1] Random Forest 과정

2.2.2 SVM

SVM 은 지도학습(Supervised Learning) 기반의 분류 알고리즘으로, 서로 다른 클래스를 나눌 때 그 여백(Margine)이 가장 넓어지는 초평면(Hyperplane)을 찾는 것을 목표로 한다[5].

SVM 은 학습 데이터 중에서 결정 경계에 가장 가까운 데이터 포인트인 서포트 벡터(Support Vector)에 의해 초평면이 정의되며, 이 서포트 벡터들은 분류 성능에 직접적인 영향을 미친다. 또한, SVM 은 단순히 선형 분리가 가능한 데이터뿐만 아니라, 커널 트릭(kernel Trick)을 이용하여 비선형적으로 분포한 데이터도 고차원 공간으로 변환하여 분류할 수 있으며 이러한 특성 덕분에 고차원 특성 공간을 가진 데이터에서도 강력한 일반화 성능을 보이며, 과적합을 효과적으로 방지할 수 있다[6].

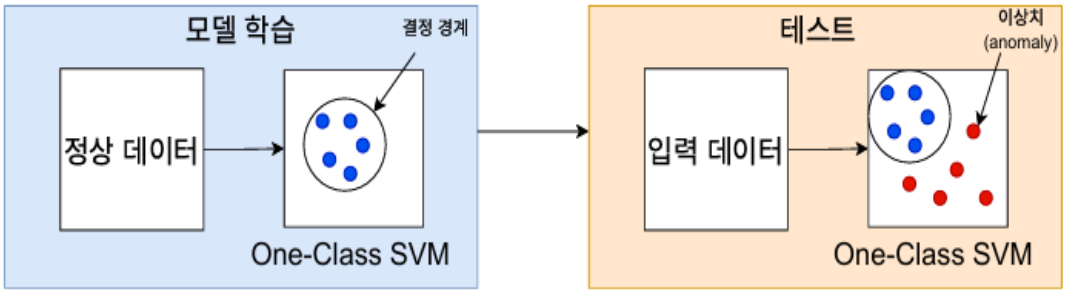


[그림 2] SVM

2.2.3 One-Class SVM

One-Class SVM 은 정상 데이터만을 학습하여 이상(anomaly)을 탐지하는 데 특화된 비지도 학습 기반 모델이다. 전통적인 SVM 이 서로 다른 두 클래스를 구분하는 초평면을 찾는 것을 목표로 하는 반면, One-Class SVM 은 하나의 클래스를 감싸는 결정 경계(decision boundary)를 학습하여, 이 경계 밖에 위치한 데이터를 이상치로 판별한다[7].

본 연구에서는 일부 악성코드 패밀리를 정상으로 간주하고 학습에 사용하여, 정상 경계 밖에 위치하는 데이터가 이상치로 탐지되는지 확인하였다.



[그림 3] One-Class SVM 과정

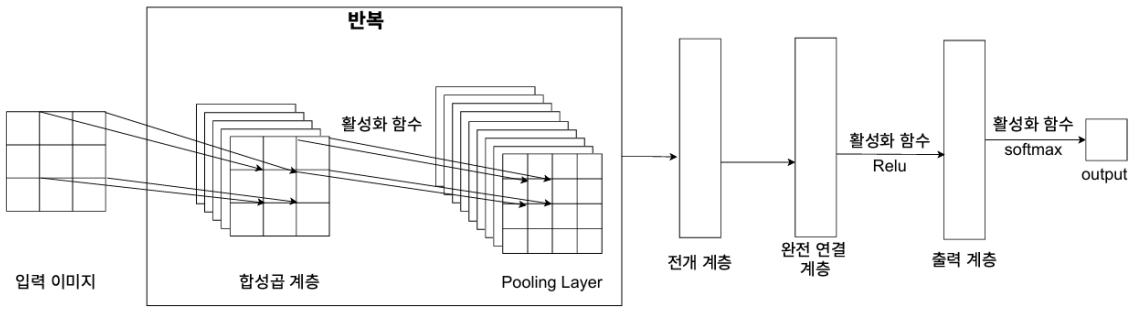
2.3 딥러닝

딥러닝(Deep Learning)은 인간 두뇌 구조를 모방한 인공신경망(Artificial Neural Network)을 기반으로 하는 머신러닝 기술로, 이미지, 음성, 자연어 등 비정형 데이터 처리에 뛰어난 성능을 보여준다.

이 중 Convolutional Neural Network (CNN)는 딥러닝에서 가장 널리 사용되는 구조 중 하나로, 합성곱 계층(Convolution Layer), 활성화 함수, 풀링 계층(Pooling Layer), 완전연결 계층(Fully Connected Layer) 등으로 구성된다[8].

합성곱 계층은 입력 이미지로부터 국소적인 특징을 추출하고, 활성화 함수는 비선형함수를 사용하여 모델이 복잡한 패턴을 학습할 수 있도록 한다. 풀링 계층은 데이터 차원을 축소시켜 계산 효율성과 공간적 불변성을 확보하며, 완전연결 계층은 추출된 특징을 기반으로 최종 분류를 수행한다.

CNN 은 이러한 계층 구조를 반복적으로 쌓아가며 점진적으로 고차원 특징을 학습하며, 이미지 분류, 객체 인식 등 다양한 문제에서 뛰어난 성능을 보인다.



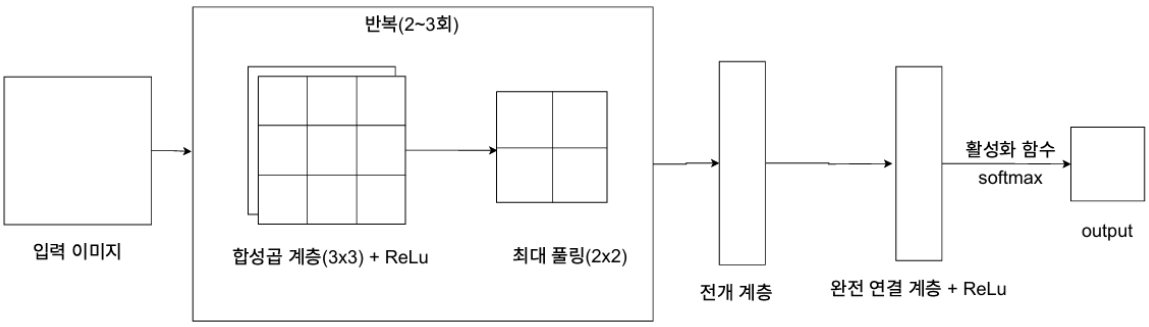
[그림 4] CNN 과정

2.3.1 VGGVisual Geometry Group

Visual Geometry Group (VGG)는 2014 년 ILSVRCImageNet Large Scale Visual Recogn 대회에서 제안된 CNN 구조로, 간결하면서 깊이 있는 모델 설계를 통해 모델의 성능을 향상시키는데 중점을 둔 모델이다[9].

VGG 의 가장 큰 특징은 동일한 크기의 작은 필터와 2x2 크기의 최대 풀링(Max Pooling)을 여러 층에서 반복적으로 적용하여 깊은 신경망 구조를 구성한다는 점이다.[2]

이러한 설계는 구조적 일관성과 구현의 단순성을 제공하면서, 계층이 깊어질수록 더욱 복잡하고 추상적인 특징을 효과적으로 추출할 수 있게 한다.



[그림 5] VGG 과정

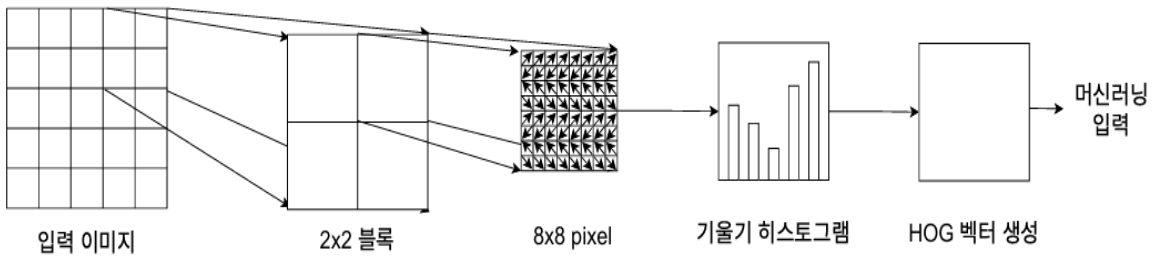
2.4 HOG Histogram of Oriented Gradients

HOG (Histogram of Oriented Gradients)는 이미지에서 객체의 형태나 경계를 효과적으로 추출하기 위해 사용되는 특징 추출 기법이다[10]. [3]

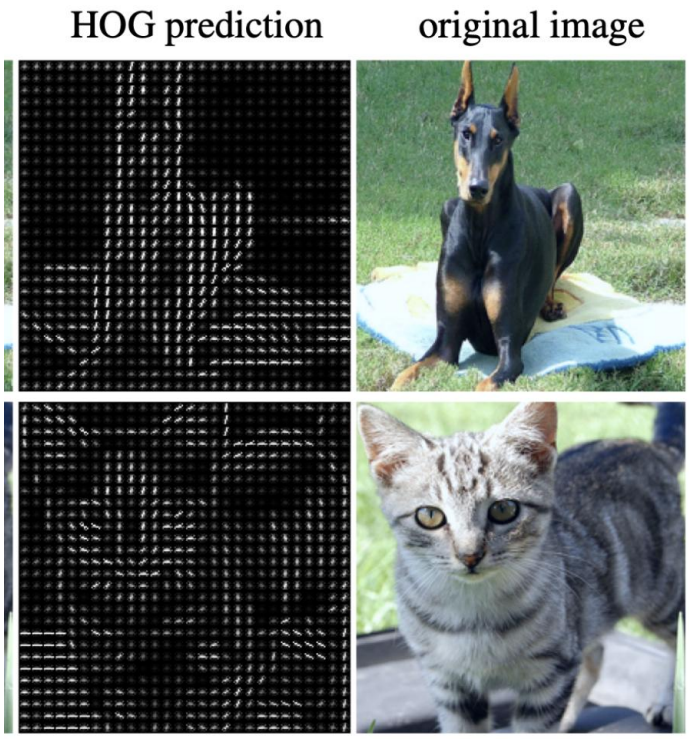
이 기법은 국소 영역(Local Region) 내에서 밝기 변화의 방향(Gradient Orientation)을 계산하고, 이를 히스토그램 형태로 정리하여 이미지의 구조적인 정보를 수치화 한다.

구체적으로는 입력 이미지를 일정 크기의 셀(Cell)로 분할한 후, 각 셀 내에서 각 화소의 수직 및 수평 방향의 기울기(Gradient)를 계산하고, 그 방향을 기준으로 히스토그램을 구성한다. 이후 여러 개의 셀을 하나의 블록으로 묶어 정규화를 수행함으로써 조명 변화나 명암 차이에 강인한 특징 벡터를 생성할 수 있다.

본 연구에서는 머신러닝 모델의 입력으로 HOG 를 적용하여, CNN 모델과의 성능을 비교하였다.



[그림 6] HOG 과정



[그림 7] HOG 결과 예시[10]

제 3 장 실험 방법

3.1 데이터셋 구성 및 실험 방식

본 연구에서는 기계학습 및 딥러닝 기반의 악성코드 탐지 성능을 비교하기 위해 Malimg 이미지 데이터셋을 활용하였다.

Malimg 데이터셋은 총 25 개의 악성코드 패밀리로 구성되어 있으며, 9,339 개의 PE(Portable Executable) 기반 악성코드 샘플을 포함한다. 각 샘플은 원본 PE 파일을 8 비트 단위의 바이트 배열로 변환한 뒤, 고정된 너비를 기준으로 2 차원 배열로 재구성하여 그레이스케일 이미지로 표현된다.

전체 데이터셋은 사전에 약 9:1 의 비율로 분할되었으며, 8,404 개의 학습용(Train) 데이터와 935 개의 검증용(Validation) 데이터로 구성되었다. 또한, 924 개의 테스트 데이터는 이와 별도로 구성하여 모델의 최종 성능 평가에만 활용하였다.

전처리 과정에서는 모든 이미지에 대해 픽셀 값을 255 로 나누는 방식의 정규화를 적용하여, 값의 범위를 0~1 로 조정하였다. 이는 모델의 학습 안정성과 수렴 속도 향상을 위한 조치이다. 이후, 모델의 구조에 따라 입력 해상도를 달리 설정하였다. 머신러닝 기반 모델(Random Forest, SVM, One-Class SVM)의 경우, 이미지 데이터를 64×64 해상도로 리사이징한 후 HOG(Histogram of Oriented Gradients) 특징 벡터를 추출하여 입력으로 사용하였다. 반면, G 모델은 이미지의 공간적 특징을 보다 풍부하게 학습할 수 있도록 256×256 해상도로 리사이징된 이미지를 그대로 입력으로 활용하였다.

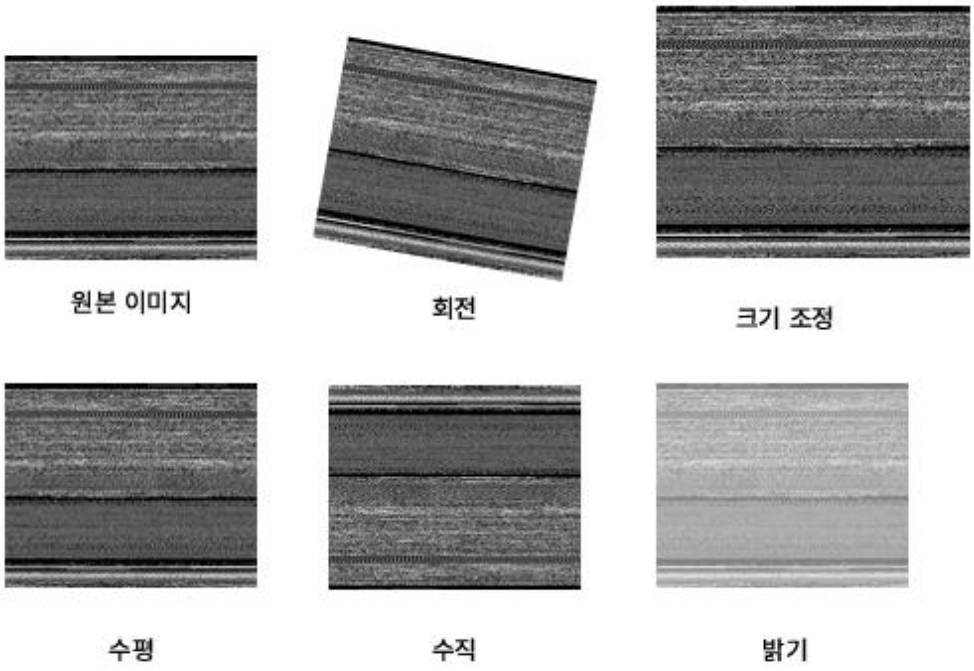
One-Class SVM 은 정상 클래스의 데이터만을 기반으로 학습하는 이상 탐지 방식이다. 그러나 Malimg 데이터셋은 전부 악성코드로만 구성되어 있기 때문에, 악성코드 패밀리 중 Allaple.A, Allaple.L, Lolyda.AA1, Lolyda.AA2 를 정상 데이터로 간주하여 학습에 사용하였다. 이는 실험적 목적의 가정이며, 실제 정상 데이터와는 차이가 있으므로 모델 성능 해석 시 유의가 필요하다.

3.2 데이터 전처리

본 연구에서는 악성코드 이미지 데이터를 활용한 모델 학습을 위해, 각 모델의 구조와 특성에 맞는 전처리 과정을 별도로 설계하였다.

전처리 단계는 크게 이미지 정규화, 리사이징, 특징 추출의 세 가지 요소로 구성되며, 딥러닝 기반 모델과 머신러닝 기반 모델에서 서로 다른 방식이 적용되었다.

딥러닝 기반 모델은 이미지 데이터를 256×256 해상도로 리사이징하였다. 이는 원본 Malimg 이미지(512×410)의 공간적 구조를 보존하면서 학습 속도와 연산 자원 사용의 효율성을 확보하기 위함이다.
모델의 일반화 성능을 향상시키고 과적합을 방지하기 위해, 학습 데이터에는 다양한 이미지 증강 기법을 적용하였다[11].[4]



[그림 8] 데이터 증강 예시

적용된 증강 기법으로는 다음과 같다.

- 최대 ±15 도 범위의 이미지 회전
- 수평 및 수직 방향으로 최대 10% 범위의 위치 이동
- 최대 10% 범위의 무작위 확대 및 축소
- 좌우 반전(대칭) 처리

이러한 증강 기법을 통해, 방향이나 위치, 크기가 달라진 악성코드 이미지에 대해서도 모델이 일관되게 패턴을 인식할 수 있도록 하였다.

```
[ 데이터 증강 기법 설정 ]
입력 이미지에 대해 다음과 같은 전처리 및 증강 기법을 적용:
  픽셀 정규화:
    모든 이미지의 픽셀 값을 255로 나누어 [0, 1] 범위로 조정
  회전:
    이미지를 최대 ±15도 범위 내에서 무작위 회전
  크기 조절:
    이미지를 최대 ±10% 범위 내에서 화대 혹은 축소
  수평 반전:
    이미지를 좌우로 무작위 반전
  위치 이동:
    가로 방향으로 ±10% 이동
    세로 방향으로 ±10% 이동
```

[그림 9] 데이터 증강 기법 의사코드

머신러닝 모델은 입력 이미지로부터 고정된 차원의 특징 벡터를 필요로 하기 때문에, 원본 이미지를 64x 64 해상도로 리사이징한 후 HOG 특징을 추출하였다.

추출된 HOG 벡터는 머신러닝 모델의 입력으로 사용되며, 이는 이미지의 시각적 형태를 수치화하여 모델이 악성코드의 패턴을 학습할 수 있도록 한다.

본 연구에서는 'scikit-image' 라이브러리의 'hog()' 함수를 활용하여 각 이미지의 특징을 벡터화 하였고[12], 이때 셀 크(8x8), 블록 크기(2x2)를 설정하여 특징을 추출하고, 추출된 결과는 하나의 일렬 벡터로 평탄화(flatten)하여 모델의 입력으로 활용하였다.

```
[ HOG 기반 전처리 과정 ]
1. 이미지 파일을 흑백(greyscale)으로 로드
2. 입력 이미지를 지정된 크기(64 x 64)로 리사이즈
3. HOG 특징 추출 실행:
    입력 이미지
    픽셀의 셀 크기 = 8x8
    블록 크기 = 2x2
    출력 형태 = 일렬 벡터 (Flatten)
```

[그림 10] HOG 기반 이미지 전처리 의사 코드

3.3 모델 별 학습 구조 및 방법

본 절에서는 각 모델의 학습 구조와 학습 방법에 대해 기술한다. 세부적인 모델 설명은 제 2 장에서 이미 다루었으므로, 본 절에서는 학습 절차 및 구성 방식에 중점을 두어 간략히 서술한다.

3.3.1 머신러닝 기반 모델 학습 방법

머신러닝 기반 모델(Random Forest, SVM, One-Class SVM)은 이미지로부터 추출한 HOG 특징 벡터를 입력으로 받아 학습을 수행하였다.

Random Forest 는 총 100 개의 결정 트리를 사용하여 앙상블 학습을 수행하였다. 이는 트리의 개수가 너무 적을 경우 과소적합(underfitting)의 위험이

있으며, 반대로 너무 많을 경우 계산 비용이 과도하게 증가하기 때문에, 일반적으로 널리 사용되는 값인 100 으로 설정하였다.

SVM 모델은 비선형 데이터 분류에 적합한 Radial Basis Function(RBF) 커널을 사용하였으며, 규제 파라미터는 C=10, 커널 계수는 gamma=0.001 로 설정하였다. C 는 오차 허용 범위를 조절하는 파라미터로, 값이 클수록 학습 데이터에 민감하게 반응하고 복잡한 결정 경계를 생성한다. gamma 는 데이터 포인트의 영향 범위를 결정하며, 작은 값을 사용할 경우 더 부드럽고 일반화된 경계가 형성된다.

한편, 위 두 모델 모두 학습 데이터의 클래스 간 불균형 문제를 고려하여 class_weight 파라미터를 'balance'로 설정하였다. 이를 통해 각 클래스의 샘플 수에 비례하여 가중치를 자동 조정하고, 소수 클래스에 대한 민감도를 유지하며 학습이 이루어지도록 하였다.

```
models = {
    "RandomForest": RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced'),
    "SVM": SVC(kernel="rbf", C=10, gamma=0.001, class_weight='balanced')
}
```

[그림 11] Random Forest 와 SVM 모델

One-Class SVM 모델은 악성코드 중 4 개의 패밀리를 정상 데이터로 간주하여 학습하였다.

먼저 전체 학습 데이터에서 해당 패밀리에 속하는 샘플만을 필터링하여 학습에 사용하였고, 모델은 sklearn.svm.OneClassSVM 을 기반으로 구현되었다.

커널은 RBF 커널을 사용하였으며, 이상치 비율을 제어하는 파라미터 nu 는 10%로 설정하였다.

또한 gamma 는 'scale'로 지정하여 입력 데이터의 분산 및 특성 수에 따라 자동으로 계산되도록 하였다[13].

```
[ gamma = 'scale' 공식 ]
gamma = 1 / (n_features * X.var) // gamma = 1 / (특징 수 x 전체 데이터 분산)
```

[그림 12] gamma 연산

```
# === 정상 클래스 선택 ===
normal_classes = ['Allaple.A', 'Allaple.L', 'Lolyda.AA1', 'Lolyda.AA2']
# === 정상 클래스 인덱스 필터링 ===
normal_indices = np.isin(y_train_all, normal_classes)
X_train_norm = X_train_all[normal_indices]
# === One-Class SVM 학습 ===
print("[Training One-Class SVM]")
ocsvm = OneClassSVM(kernel='rbf', nu=0.1, gamma='scale')
ocsvm.fit(X_train_norm)
```

[그림 13] One-Class SVM 모델

3.3.2 딥러닝 기반 모델 학습 방법

딥러닝 기반 악성코드 탐지 모델은 VGG 스타일의 합성곱 신경망(CNN) 구조를 기반으로 설계되었다. 이 모델은 입력된 이미지를 통해 자동으로 특징을 추출하고, 이를 바탕으로 각 악성코드 패밀리를 분류한다.

입력 데이터는 256×256 해상도의 단일 채널 이미지로 전처리되며, 모델의 입력 계층은 (256, 256, 1) 형태의 텐서를 사용한다.

모델은 계층이 깊어질수록 점점 더 복잡하고 추상적인 특징을 학습할 수 있도록 구성되어 있다. 초기 계층에서는 경계나 질감 등 국소적인 특징을 인식하고, 이후 계층에서는 악성코드의 형태적 패턴이나 구조와 같은 고차원적인 정보를 학습하게 된다.

출력 계층에는 softmax 함수를 적용하여 Malimg 데이터셋의 25 개 클래스에 대한 다중 클래스 분류를 수행한다

```
model = Sequential([
    Input(shape=(256, 256, 1)),

    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Conv2D(256, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(25, activation='softmax')  # Malimg 데이터셋은 25개 클래스
])
```

[그림 14] VGG 모델 구조

손실 함수는 categorical_crossentropy 를[14] 사용하였으며, 최적화 기법으로는 Adam 을 적용하였다[15].

학습률은 0.0001 로 설정하여 안정적인 수렴을 유도하였고, 총 20 에폭(epoch) 동안 학습을 진행하였다.

```
[ VGG모델 컴파일 ]
모델 컴파일:
  최적화 함수 = Adam (학습률 = 0.001)
  손실 함수 = categorical_crossentropy
  평가 지표 = accuracy
```

[그림 15] VGG 모델 컴파일

제 4 장 테스트 및 성능 평가

4.1 성능 평가 개요

본 논문에서는 Malimg 데이터 셋을 이용하여 악성코드 탐지 모델들을 설계 및 구현을 하였다. 본 장에서는 구현한 모델들을 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1-score, 그리고 Area Under the Curve (AUC)를 기준으로 평가하고 비교하여 최선의 모델을 찾아내고자 한다[16].

지표 이름	**수식**	**설명**
Accuracy (정확도)	(TP + TN) / (TP + TN + FP + FN)	전체 샘플 중에서 올바르게 분류된 비율
Precision (정밀도)	TP / (TP + FP)	모델이 양성이라고 예측한 것 중 실제 양성의 비율
Recall (재현율)	TP / (TP + FN)	실제 양성 중 모델이 양성으로 올바르게 예측한 비율
F1-score	2 × (Precision × Recall) / (Precision + Recall)	정밀도와 재현율의 조화 평균으로, 두 지표 간 균형을 평가하는 지표
AUC	$\mathrm{AUC} = \int_0^1 TPR(FPR)\, dFPR$	클래스별 분류 민감도를 종합적으로 평가하는 지표

[표 2] 성능 평가 지표

TP(True Positive), TN(True Negative), FP(False Positive), FN(False Negative)는 모델의 예측 결과와 실제 정답 간의 관계를 나타내는 분류 결과 지표이다.
이로부터 도출되는 TPR(True Positive Rate)과 FPR(False Positive Rate)은 모델의 판단이 실제 정답과 얼마나 일치하는지를 나타내는 대표적인 성능 지표로,
양성 클래스에 대한 예측 결과에서 참(정답)과 거짓(오답)의 비율을 수치화한 것이다[16].

4.2 성능 테스트 및 분석

본 절에서는 앞서 학습한 네 가지 모델을 대상으로 Malimg 검증 데이터셋을 사용하여 성능을 평가하였다. 평가 지표로는 정확도, 정밀도, 재현율, F1-score 을 사용하였으며, 각 모델의 성능은 [표 3]와 같다.

Model	**정확도**	**정밀도 (Macro)**	**재현율 (Macro)**	**F1-score (Macro)**	**AUC**
Random Forest	0.917	0.86	0.86	0.86	0.8773
SVM	0.897	0.81	0.83	0.81	0.8894
One-Class SVM	0.920	0.93	0.88	0.90	–
CNN (VGG)	0.980	0.97	0.97	0.96	0.9119

[표 3] 모델 성능 평가

실험 결과, CNN(VGG) 모델은 다중 클래스 분류에서 98%의 높은 정확도를 기록하며, 모든 지표에서 가장 우수한 성능을 나타냈다. AUC 측면에서도 0.9119 로 가장 높은 값을 기록하여, 클래스 간 분류 민감도 면에서도 뛰어난 구분 능력을 보였다.

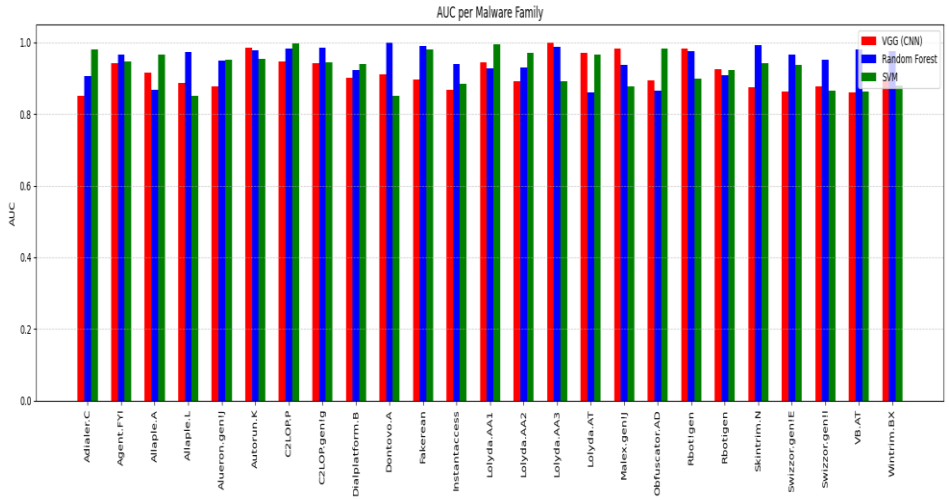
머신러닝 기반 모델 중에서는 Random Forest 가 91.7%의 정확도와 0.8773 의 AUC 로, 안정적이고 균형 잡힌 분류 성능을 보였다.

SVM 모델은 AUC 가 0.8894 로 Random Forest 보다 약간 높았지만, 정확도 및 F1-

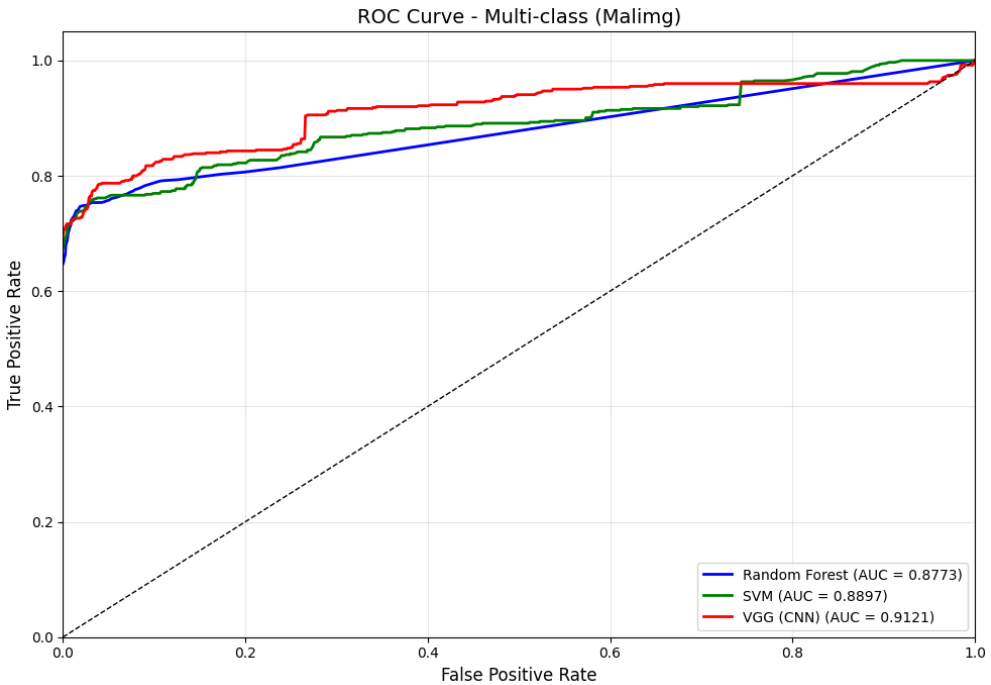
score 측면에서는 소폭 낮은 수치를 나타냈다.

한편, One-Class SVM 은 정상 데이터를 기반으로 학습된 이상 탐지 모델로서, 전체 악성코드 중 비정상적 패턴을 효과적으로 식별하며 안정적인 탐지 성능을 확인하였다.

단, 이 모델의 특성상 정밀한 클래스 분류보다는 이상 탐지 관점에서 해석해야 하며, AUC 계산은 적용되지 않았다.



[그림 16] 각 패밀리별 AUC 막대그래프



[그림 17] ROC 곡선과 AUC 비교 그래프

제 5 장 결론

본 논문에서는 머신러닝 및 딥러닝 기법을 활용하여 악성코드 탐지모델을 구현하고 성능을 비교 분석하였다. 실험에 사용된 Malimg 데이터셋은 PE 기반 악성코드 샘플을 2 차원 그레이스케일 이미지로 변환한 형태로, 이미지 처리 기반의 탐지 모델 적용 가능성을 제시하였다.

머신러닝 기반의 탐지 모델로는 Random Forest, SVM, One-Class SVM 을 사용하였으며, 각 모델은 HOG 특징 벡터를 입력으로 받아 학습을 수행하였다. 반면, 딥러닝 기반의 CNN(VGG) 모델은 원본 이미지 데이터를 입력으로 하여 자동으로 특징을 추출하고 학습을 수행하였다.

성능 평가 결과, CNN 기반의 VGG 모델이 정확도(98%), 정밀도(97%), 재현율(97%), F1-score(96%) 등 모든 지표에서 가장 높은 성능을 기록하였으며, 특히 다중 클래스 분류 문제에 있어 뛰어난 일반화 능력을 보였다. 또한, 다중 클래스 분류 문제에 대한 민감도 평가 지표인 AUC 분석 결과에서도 VGG 모델이 가장 높은 값을 기록하며, 전반적인 분류 성능 우수성을 입증하였다. 머신러닝 기반 모델 중에서는 Random Forest 가 전반적으로 안정적인 성능을 나타냈으며, One-Class SVM 은 정상 데이터로 설정한 패밀리를 기반으로 이상 탐지에 유효한 결과를 도출하였다. 본 논문은 악성코드 탐지 분야에서 머신러닝과 딥러닝 모델의 효과성을 비교하였고 특히 CNN CNN 기반 모델이 악성 코드에 대해 높은 탐지 성능을 보인다는 점은 향후 실시간 악성코드 탐지 시스템의 기반 기술로 활용 가능성을 제시한다.

그러나 본 연구에는 몇 가지 한계점이 존재한다. 첫째, 사용된 Malimg 데이터셋은 고정된 변환 규칙과 이미지 해상도를 가지므로, 실제 환경에서의 다양한 악성코드 변형에 대한 일반화 성능을 충분히 검증하기 어렵다. 둘째, One-Class SVM 은 일부 악성코드 패밀리를 정상으로 간주하여 학습하였기 때문에 실환경의 정상 데이터 분포를 완전히 반영하기는 어렵다. 셋째, CNN 기반 모델은 높은 성능을 기록했지만, 연산 자원이 제한된 환경에서는 실시간 적용에 어려움이 있을 수 있다.

향후 연구에서는 다양한 유형의 악성코드 데이터를 확보하여 보다 현실적인 실험 환경을 구성하고, 경량화 된 딥러닝 구조나 자원 효율적인 탐지 모델을 도입함으로써, 실시간 악성코드 탐지 시스템으로의 실용성을 한층 더 높여 나가야 할 것이다.

참고 문헌

[1] "머신러닝(ML)이란?", 2025-05-18,

https://cloud.google.com/learn/what-is-machine-learning?hl=ko

[2] "[쉽게 만나는 IT] 공격 탐지 방식의 차이로 알아보는 안전한 웹방화벽",

2020-09-09, https://blog.naver.com/pentamkt/222084745829

[3] "랜덤 포레스트란 무엇인가요?", 2025-05-18

https://www.ibm.com/kr-ko/think/topics/random-forest

[4] "[머신러닝] 앙상블- 투표(Majority Voting)/

배깅(Bagging)/랜덤포레스트(Random Forest)/부스팅(Boosting)",

2021-03-08, https://bigdaheta.tistory.com/32

[5] "[머신러닝] 서포트벡터머신(SVM) 개념편(1): linear SVM", 2020-12-31,

https://blog.naver.com/chunsa0127/222190970229

[6] "[머신러닝] 서포트벡터머신(SVM) 개념편(2): kernel trick", 2021-01-06,

https://m.blog.naver.com/chunsa0127/222198868593

[7] "One-class Support Vector Machine(1-SVM)에 대하여 알아보자 with Python", 2023-05-12,

https://zephyrus1111.tistory.com/468

[8] "[딥러닝] CNN - 특징 추출 과정", 2022-06-29,

https://velog.io/@yeonheedong/DL-CNN-%ED%8A%B9%EC%A7%95-%EC%B6%94%EC%B6%9C-%EA%B3%BC%EC%A0%95

[9] "VGG-Net Architecture Explained", 2022-06-29,

https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f

[10] "Histograms of Oriented Gradients (HOG)", 2024-03-08,

https://velog.io/@hsbc/Histograms-of-Oriented-Gradients-HOG

[11] "데이터 증강이란 무엇인가요?", 2024-05-07,

https://www.ibm.com/kr-ko/think/topics/data-augmentation

[12] "HOG feature extraction – scikit-image documentation", 2025-05-01,

https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html

[13] "서포트 벡터 머신(SVM, Support Vector Machine) [내가 공부한 머신러닝 #21.]", 2019-12-02,

https://blog.naver.com/PostView.naver?blogId=gdpresent&logNo=221723178689

[14] "다중 분류 손실함수(categorical_crossentropy)와 sparse_categorical_crossentropy 차이", 2022-07-12,

https://bigdaheta.tistory.com/65

[15] "Optimization 최적화 기법", 2020-01-13,

https://my-coding-footprints.tistory.com/101

[16] "[AI] 분류 성능 평가 지표", 2023-08-24,

https://velog.io/@acadias12/AI-%EB%B6%84%EB%A5%98-%EC%84%B1%EB%8A%A5-%ED%8F%89%EA%B0%80-%EC%A7%80%ED%91%9C

감사의 글

순천향대학교 컴퓨터공학과에 입학해 정신없이 공부하다 보니, 어느덧 졸업을 앞둔 4 학년이 되었습니다. 컴퓨터공학에 대해 아무것도 몰랐던 제가 이렇게 성장할 수 있었던 것은 모두 여러 교수님들의 가르침 덕분이라고 생각합니다.

하상호 교수님, 천인국 교수님, 홍인식 교수님, 이상정 교수님, 이해각 교수님, 남윤영 교수님, 조용원 교수님을 비롯한 많은 교수님들께서 열정적으로 강의해 주신 덕분에, 저는 항상 즐거운 마음으로 수업에 임할 수 있었습니다. 진심으로 감사드립니다.

그리고 지난 대학 생활 동안 함께해 준 20 학번 친구들에게도 고마운 마음을 전하고 싶습니다.

항상 고마웠고, 덕분에 정말 즐거운 대학 생활을 보낼 수 있었습니다.

감사합니다.