


캡스톤 디자인 I 종합설계 프로젝트

프로젝트 명	<i>asi(a security insight)</i>
팀 명	<i>assist(a security safety important special team)</i>
문서 제목	결과보고서

Version	1.0
Date	2020-06-06

팀원	손 현기 (조장)
	김 주환
	김 호준
	오 예린
	이 동윤
	RUSLAN
지도교수	윤 명근 교수

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “asi(a security insight)”를 수행하는 팀 “assist(a security safety important special team)”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “assist(a security safety important special team)”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	중간보고서-결과보고서-asi(a security insight).doc
원안작성자	김주환
수정작업자	

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2020-06-06	김주환	1.0	최초 작성	초안 작성

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

목 차

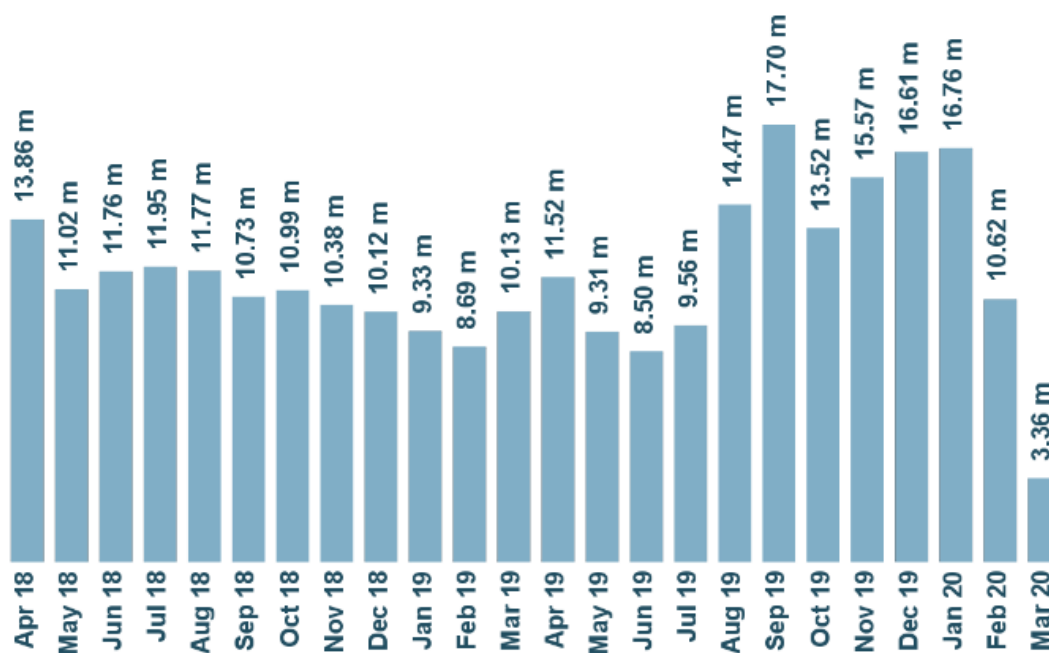
1	개요.....	4
1.1	프로젝트 개요	4
1.2	추진 배경 및 필요성	7
1.2.1	국내외 악성코드 분석 연구 동향.....	7
1.2.2	국내외 시장 현황	13
1.2.3	국내외 경쟁기관 현황.....	14
1.2.4	딥러닝 기반 악성코드 분석 보조도구의 필요성	19
2	개발 내용 및 결과물	21
2.1	목표	21
2.2	연구/개발 내용 및 결과물	23
2.2.1	연구/개발 내용	23
2.2.2	시스템 기능 요구사항.....	37
2.2.3	시스템 비기능(품질) 요구사항	39
2.2.4	시스템 구조 및 설계도.....	39
2.2.5	활용/개발된 기술	41
2.2.6	현실적 제한 요소 및 그 해결 방안.....	44
2.2.7	결과물 목록	44
2.3	기대효과 및 활용방안	45
3	자기평가.....	46
4	참고 문헌.....	47
5	부록.....	49
5.1	사용자 매뉴얼	49
5.2	테스트 케이스	49
5.3	임베딩 알고리즘별 실험 결과	50

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

1 개요


1.1 프로젝트 개요

정보보안 전문 기업 AV-Test의 통계 조사에 따르면 매일 약 350000 개의 악성코드가 새로 생성되고 있으며, 2019년에는 매달 평균적으로 약 12075800 개의 악성코드가 새로 생성되었다 [1]. 매일 생성되는 수많은 악성코드를 전문가가 분석하는 것은 현실적으로 어려우므로 일반적으로 자동 분석 도구를 통해 악성코드를 분석하고, 대응하는 방법을 채택하고 있다. 전통적인 자동 분석 방안으로는 악성코드의 행위 규칙이나 해시 값을 이용하는 서명 탐지(signature detection)와 정상코드의 API(application programming interface) 규칙 기반의 이상 탐지(anomaly detection)가 있다. 그러나, 전통적인 방법은 변종 악성코드를 탐지하지 못하는 문제가 존재한다 [2, 7].



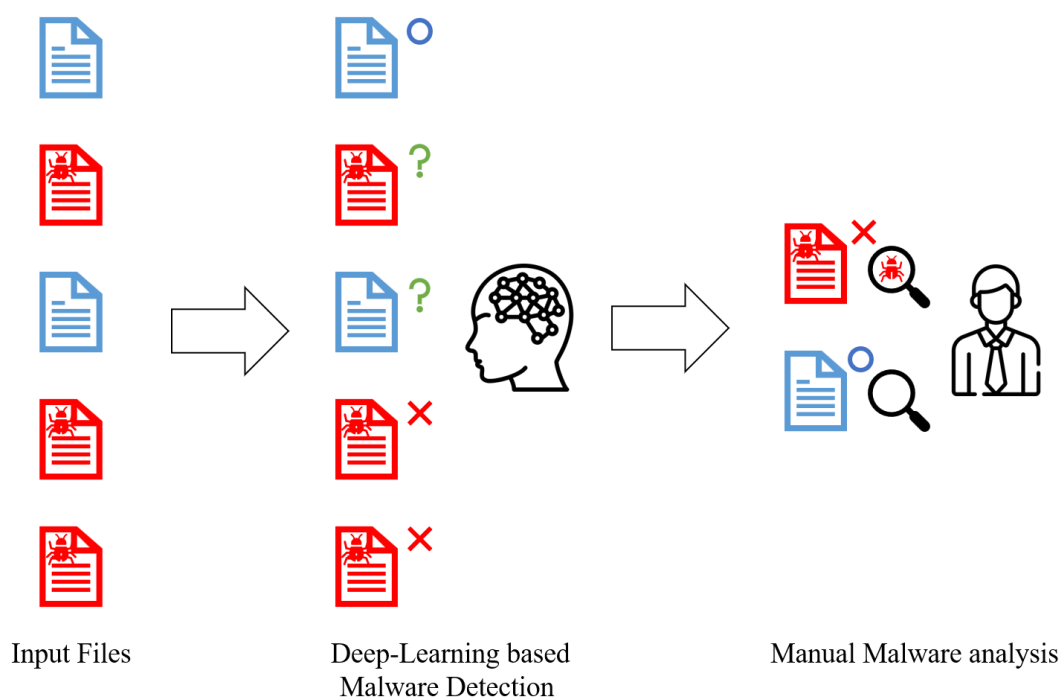
[그림 1] AV-Test의 신종 악성코드 발생량 통계조사 결과 [1]

최근에는 기계학습(machine learning) 및 딥러닝(deep learning)을 기반으로 하는 악성코드 자동 분석기 연구가 활발히 진행되고 있다 [3-6, 8-10]. 딥러닝 기반의 악성코드 분석은 데이터로부터 특징을 학습하여 분류를 수행하므로 악성코드의 변형을 감내할 수 있으며, 전통적인 분석 방법에 비해 월등히 높은 성능을 갖는다. 그러나, 전통적인 암호 분석 도구 뿐만 아니라 딥러닝 기반의 악성코드 분석기 역시 100%의 정확도로 분석할 수 없으므로, 일부 악성코드에 대해서는 여전히 전

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

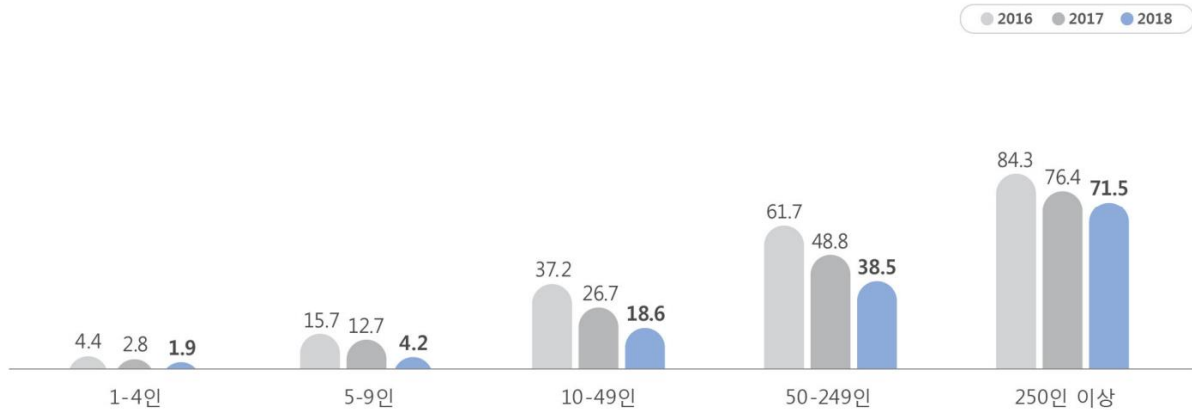
문가의 분석이 필요하다.

[그림 2]는 일반적인 악성코드 분석 방법을 도식화한 것이다. 악성파일(붉은색)과 정상파일(파란색)이 입력으로 들어왔을 때, 먼저 자동 분석기가 파일의 악성/정상 여부를 판별한다. 대부분의 파일은 이 단계에서 분류되나, 악성/정상 여부가 불분명한 일부 파일에 대해서는 분석기가 판단하지 못하고 전문가에게 분석을 요청한다. 최근 딥러닝 기반 자동 분석기의 성능은 크게 향상되고 있으나, 악성코드의 수 역시 증가하고 있는 추세이므로 여전히 전문가가 많은 악성코드를 직접 분석해야 하는 실정이다.



[그림 2] 일반적인 악성코드 분석 방법

악성코드 분석 전문가의 수요는 증가하는 반면, 국내의 정보보호에 대한 투자는 매우 빈약한 상황이다. 과학기술정보통신부의 2018년 정보보호실태조사에 따르면 국내 기업 중 정보보호 조직을 보유하지 않은 기업의 비율은 94.5%이고, 정보보호 또는 개인정보보호에 투자하지 않는 기업의 비율은 63.8%, IT 예산 중 1% 미만을 투자하는 비율은 89.1%에 달한다 [12].




[그림 3] 규모별 정보보호(개인정보보호) 조직 보유율 [12]

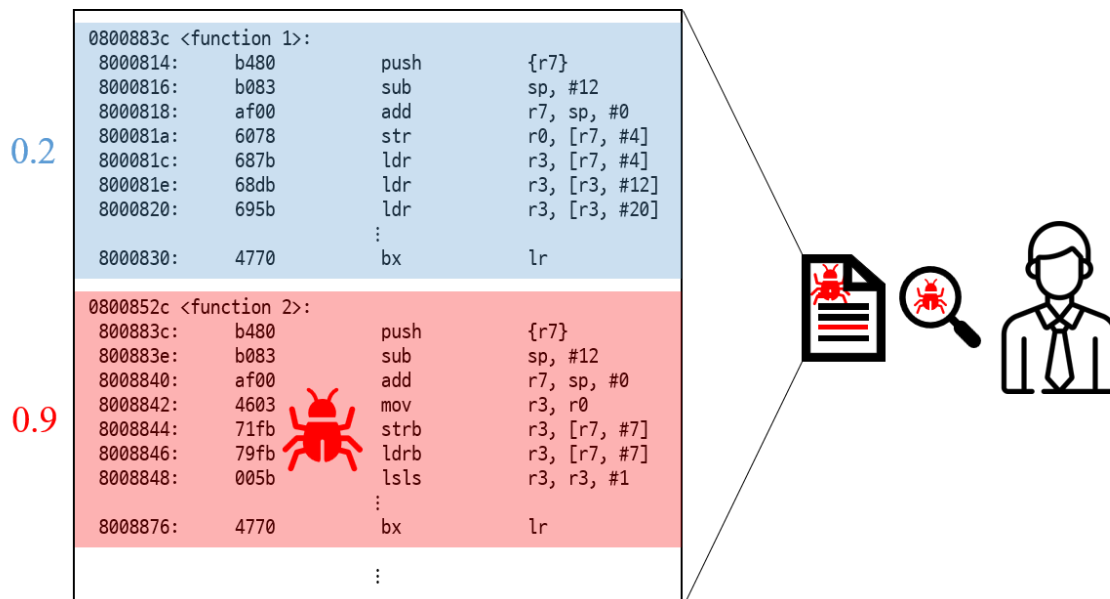


[그림 4] IT 예산 중 정보보호(개인정보보호) 예산 비중 (투자하지 않는 기업은 미표기) [2]

정보보안 전문 기업 AhnLab에 따르면, 전문가가 하나의 악성코드를 분석하는데 최소 몇 시간에서 최대 몇 주가 소요된다 [11]. 최근 악성코드의 크기가 커지면서, 전문가가 확인해야 하는 코드의 수가 증가하는 추세이므로 악성코드 분석에 소요되는 시간은 더 길어질 것으로 예상된다. 정보보안에 대한 투자가 적고 생성되는 악성코드는 많아지며, 분석에 소요되는 시간이 길어지는 현 상황에서 안전한 시스템을 구축하기 위해서는 전문가의 악성코드 분석시간을 줄일 수 있는 소프트웨어의 개발이 시급하다.

우리의 목표는 [그림 5]와 같이 입력 파일의 각 함수 별 악성행위 수행 가능성을 점수화 하는 소프트웨어를 개발함으로써 전문가의 악성코드 분석 가능성을 줄이는 것이다. 우리의 소프트웨어가 성공적으로 개발된다면, 동일한 인력으로 더 많은 악성코드를 효율적으로 분석할 수 있으므로 정보보안을 위한 비용을 절감할 수 있으며 신종 악성코드에 신속히 대응할 수 있을 것으로 기대된다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 5] 제안하는 소프트웨어의 목표

1.2 추진 배경 및 필요성

1.2.1 국내외 악성코드 분석 연구 동향


악성코드 분석 연구는 [그림 2]의 첫 번째 단계인 자동 분석기의 성능 향상에 치중되어 있고, 두 번째 단계인 전문가의 악성코드 분석 보조도구에 대한 연구는 전무한 상황이다. 본 항에서는 악성코드 분석 연구 동향을 살펴봄으로써 2장에서 기술할 우리가 제안하는 연구의 방법론과 필요성에 대한 기반 지식을 제시한다.

악성코드 분석 연구의 목적은 크게 악성/비악성 분류 [3-5], 악성코드의 패밀리 분류 [6-10]로 나눌 수 있다. 악성/비악성 분류기는 가장 기본적인 연구 목표로 입력 파일의 악성 행위 여부를 판단한다. 악성코드 패밀리 분류는 악성코드의 실행 행위에 따라 패밀리를 분류하는 것으로 적절한 방어기법 제공을 위해 필요하다. 아래의 두 목에서 각 연구 동향을 상세히 기술한다.

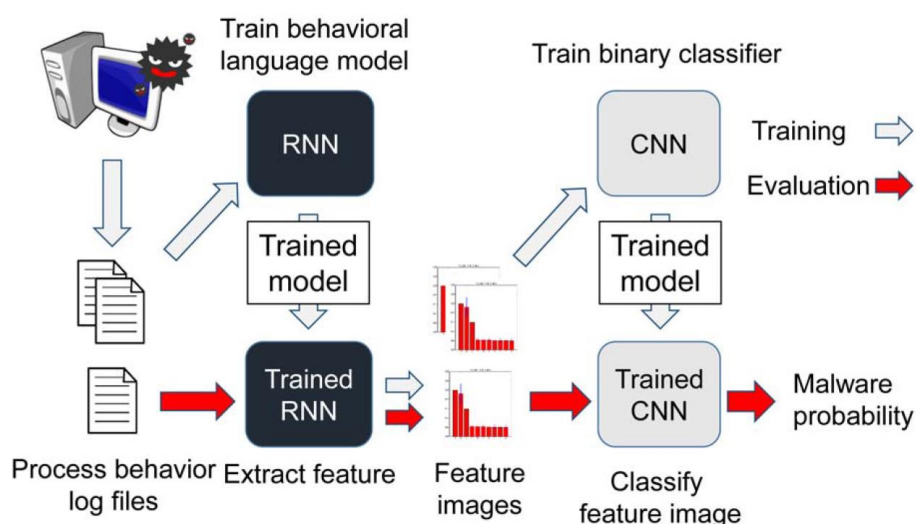
1.2.1.1. 악성코드 분류 연구 동향

● Malware Detection with Deep Neural Network Using Process Behavior [3]

Tobiyama, 등은 RNN(recurrent neural network, 순환신경망), CNN(convolution neural network, 합성곱신경망)을 기반으로 프로그램의 트래픽을 분석하여 악성파일과 정상파일을 최대 96%의 정확도로 분류하는 기법을 제안했다. 분류기는 [그림 6]과 같이 세 단계를 거쳐 분류를 수행한다. 먼저 프로그램의 트래픽을 Microsoft 사의 Process Monitor를 이용하여 트래픽 데이터를 수집한다. 다음으

 <div> <p>국민대학교</p> <p>컴퓨터공학부</p> <p>캡스톤 디자인 I</p> </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

로 RNN의 일종인 LSTM(long short-term memory, 장단기메모리)을 이용해 트래픽으로부터 특징 이미지를 생성한다. 마지막으로 특징 이미지를 CNN으로 분석하여 파일의 정상/악성 여부를 판별한다.




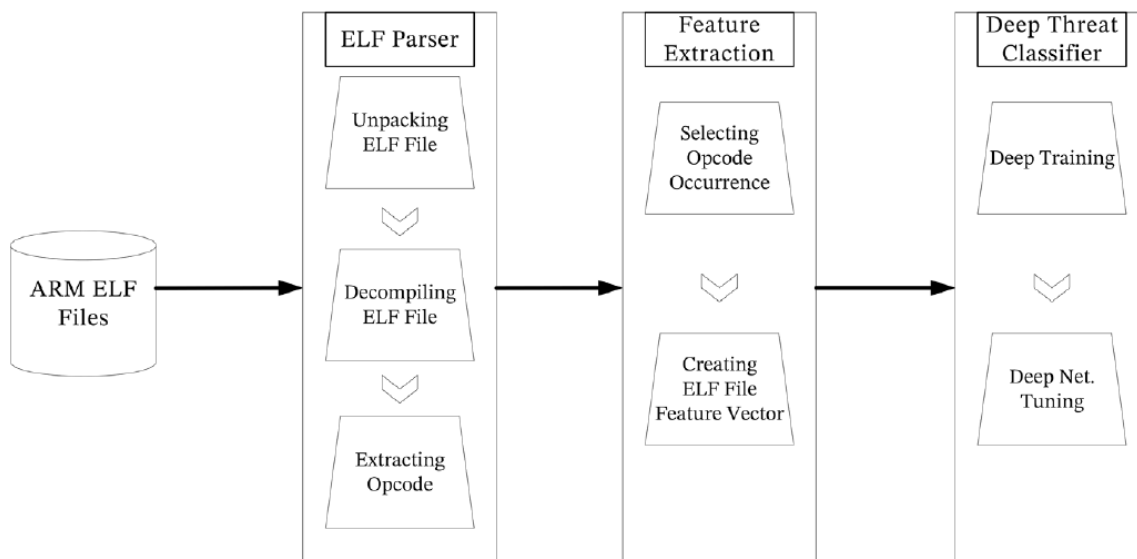
[그림 6] [3]의 악성코드 분류 방법의 개요

이 논문에서 주목해야 할 점은 두 번째 단계인 특징 추출(feature extraction) 단계이다. 저자들은 LSTM이 언어 모델(language model)을 학습할 수 있도록 현재 명령어를 입력으로 받아 다음 명령어를 예측하도록 학습시켰다. 이는 정상코드와 악성코드의 명령어의 분포가 다를 것을 시사하므로, 정상코드에 대한 명령어의 분포를 학습시킨 뒤 악성코드의 명령어의 분포를 검사한다면, 파일 중 악성 행위를 수행하는 코드를 탐지할 수 있을 것으로 기대된다.

● A Deep Recurrent Neural Network based approach for Internet of Things malware threat hunting [4]

HaddadPajouh, 등은 양방향 RNN(bidirectional RNN, BRNN)을 기반으로 프로그램의 opcode를 분석하여 악성파일과 정상파일을 최대 98.18%의 정확도로 분류하는 기법을 제안했다. 저자들은 ELF(executable and linking format) 파일에서 GNU 바이너리 유틸리티 중 disassembler인 objdump를 이용해 opcode를 추출했다. 특징의 개수를 줄이기 위해 정보이득(information gain, IG) 값을 계산하여 0.3 초과인 opcode에 대한 분석을 수행했다.

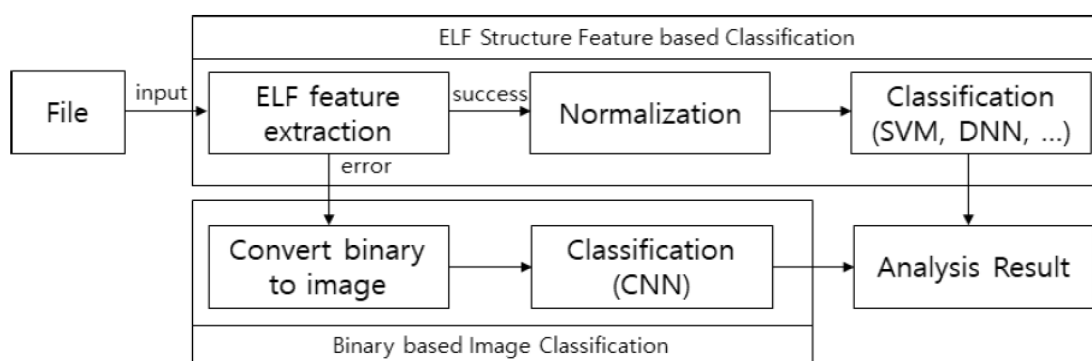
 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 7] [4]의 악성코드 분류 방법의 개요

● 정적 분석과 앙상블 기반의 리눅스 악성코드 분류 연구 [5]


황준호, 등은 SVM(support vector machine), Random Forest, naïve Bayes, k-NN(k-nearest neighbor), MLP를 기반으로 ELF 파일을 최대 99.68%의 정확도로 분석하는 기법과, CNN을 기반으로 이미지화된 바이너리 데이터를 95.05%로 분석하는 기법을 제안했다. 정상코드와 악성코드의 ELF 파일의 통계적 특성(ELF header의 크기, program header의 수, 등)이 차이가 있음을 보임으로써 ELF 파일 기반 악성코드 분석의 타당성을 보였다. 또한, ELF 파일이 존재하지 않는 파일을 CNN 기반으로 분석하는 방법을 제안했다.



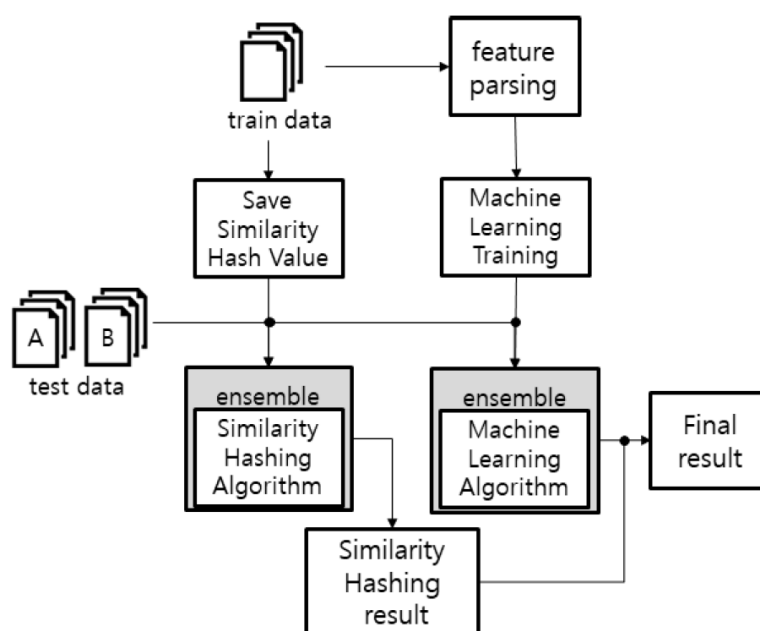
[그림 8] [5]의 악성코드 분류 방법의 개요

● 정적 분석 기반 기계학습 기법을 활용한 악성코드 식별 시스템 연구 [6]

김수정, 등은 유사성 해시 알고리즘과 기계학습 알고리즘 앙상블을 기반으로 PE 파일을 분석하여 악성파일과 정상파일을 최대 95.9%의 정확도로 분류하는 기법을 제안했다. 이 연구에서는

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

ssdeep, DHASH(difference hash), TLSH(trend micro locality sensitive hash) 세 유사성 해시 알고리즘을 앙상블하고, k-NN, Decision Tree, SVM, MLP 네 기계학습 알고리즘을 앙상블하여 정확도를 향상시켰다.




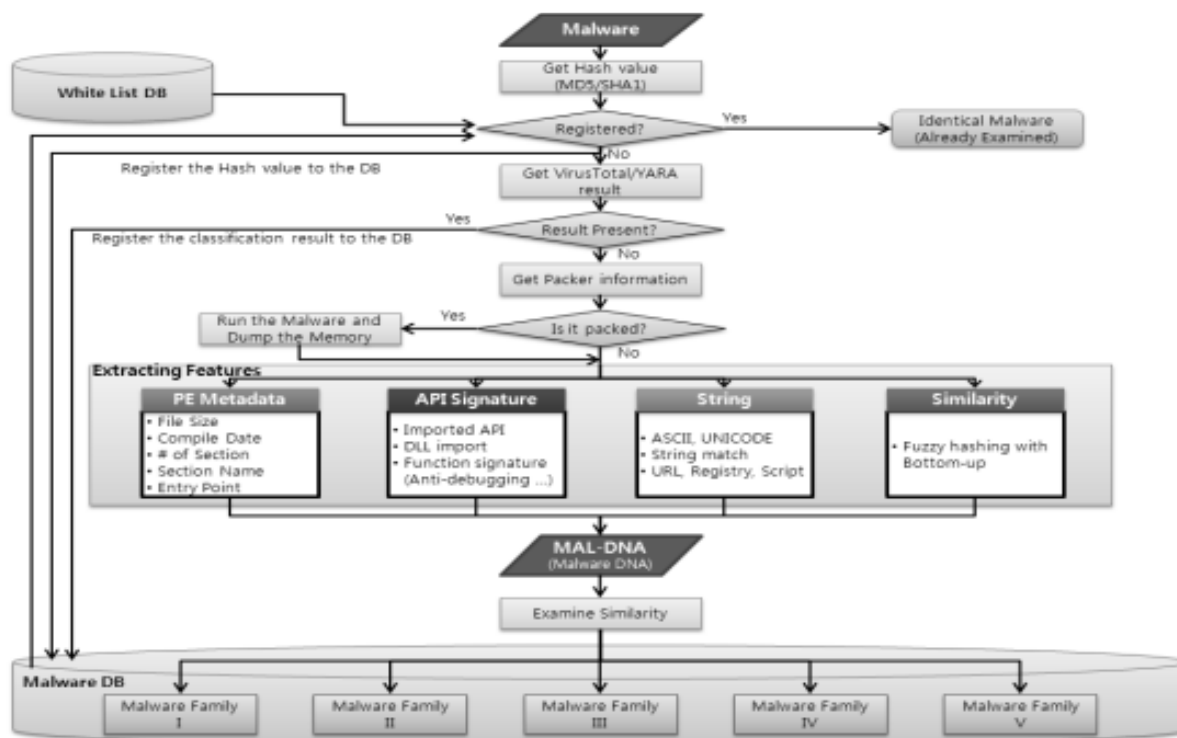
[그림 9] [6]의 악성코드 분류 방법의 개요

1.2.1.2. 악성코드 패밀리 분류 연구 동향

● 악성코드 DNA 생성을 통한 유사 악성코드 분류기법 [7]

한병진, 등은 API 호출, PE 파일의 구조 등을 규칙 기반으로 분석하여 4 개의 악성코드 패밀리를 약 83.9%의 정확도로 분류하는 기법을 제안했다. 이전의 연구에서는 악성코드의 일부 특징만을 이용하여 분류를 수행했다면, 이 연구에서는 API뿐만 아니라 다양한 특성인자를 활용하여 유사도를 검증함으로써 정확도를 높였다. 저자들은 PE(portable executable) 파일의 메타데이터, API 서명, 스트링, 등을 이용해 입력된 파일별로 악성코드 DNA (malware DNA)를 계산할 수 있는 방법론을 제안했다.

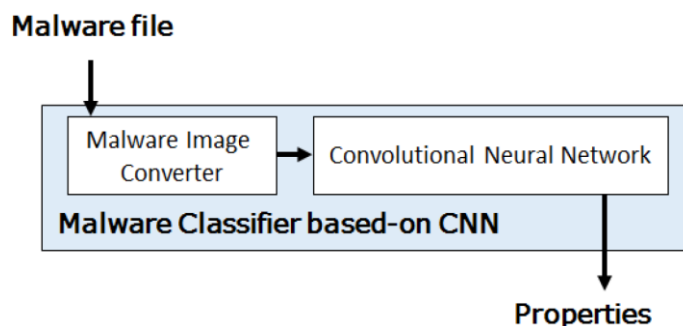
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 10] [7]의 악성코드 DNA(Malware DNA) 생성 및 분류 방법의 개요

● Convolutional Neural Network 기반의 악성코드 이미지화를 통한 패밀리 분류 [8]


석선희, 등은 CNN을 기반으로 이미지화한 바이너리 데이터를 분석하여 악성코드 패밀리의 수가 9인 경우 96.2%의 정확도, 패밀리 수가 27인 경우 82.9%의 정확도로 분류하는 기법을 제안했다. 제안된 방법은 특징 추출과 같은 별도의 전처리 없이 CNN만으로 종단간 기계학습(end-to-end machine learning)을 수행하므로 하나의 악성코드를 분류하는데 4ms의 빠른 시간 내에 분류가 가능하다는데 의의가 있다.



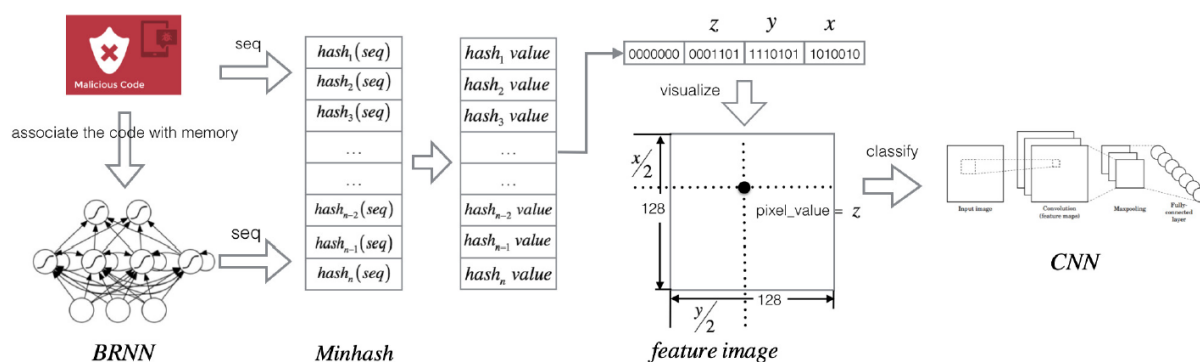
[그림 11] [8]의 악성코드 분류 방법의 개요

● Deep Learning and Visualization for Identifying Malware Families [9]

Sun, 등은 BRNN과 CNN을 기반으로 opcode를 분석하여 6 개의 악성코드 패밀리를 99.5%의 정

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

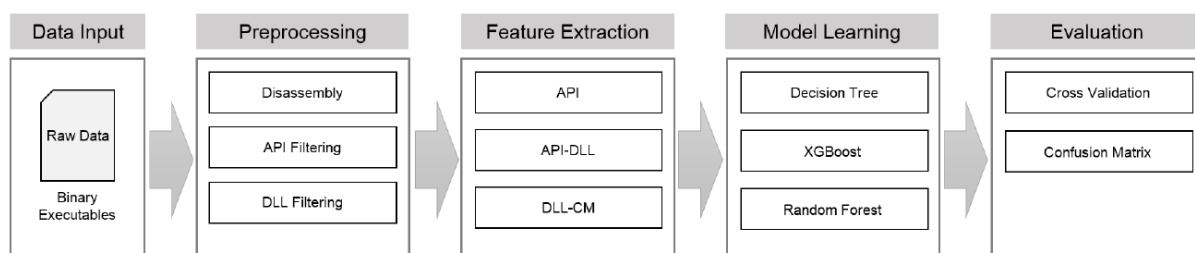
확도로 분류하는 기법을 제안했다. 제안된 기법은 BRNN을 이용해 추출한 특징과 바이너리 코드 각각에 대해 MinHash 알고리즘을 적용한 것을 결합한 후 이미지화한다. 생성된 이미지를 CNN으로 분석해 악성코드 패밀리를 분류한다. 이 연구에서는 opcode의 수를 줄이기 위해 가장 많이 나온 255개의 opcode를 남기고, 나머지 opcode는 하나의 새로운 opcode로 치환했다. LSTM의 그레디언트 소멸(gradient extinction) 문제와 그레디언트 폭발(gradient explosion) 문제를 막기 위하여 GRU(gated recurrent unit, 게이트 순환 유닛)을 이용해 BRNN을 구현했다.




[그림 12] [9]의 악성코드 분류 방법의 개요

● 악성코드 패밀리를 위한 API 특징 기반 양상블 모델 학습 [10]

이현종, 등은 XGBoost, Random Forest를 기반으로 API와 DLL(dynamic link library)를 분석하여 패밀리의 수가 2인 경우 93.6%의 정확도, 패밀리의 수가 5인 경우 93.5%의 정확도로 분류하는 기법을 제안했다. 제안된 기법은 API, API-DLL, DLL-CM 세 가지 특징을 추출하고, 각각에 대해 신경망을 학습시켰다. API는 API 함수로부터 추출한 특징이며, 차원 감쇄를 위해 사용 빈도가 0.10% 이상인 API만을 학습에 사용했다. API-DLL은 사용 빈도가 0.25% DLL으로부터 사용된 API를 추출했으며, DLL-CM은 DLL의 호출 관계를 CM(call matrix)로 표현한 것으로 DLL 호출 순서로부터 2-gram 구조를 2차원 행렬 형태로 변형하여 생성한다.

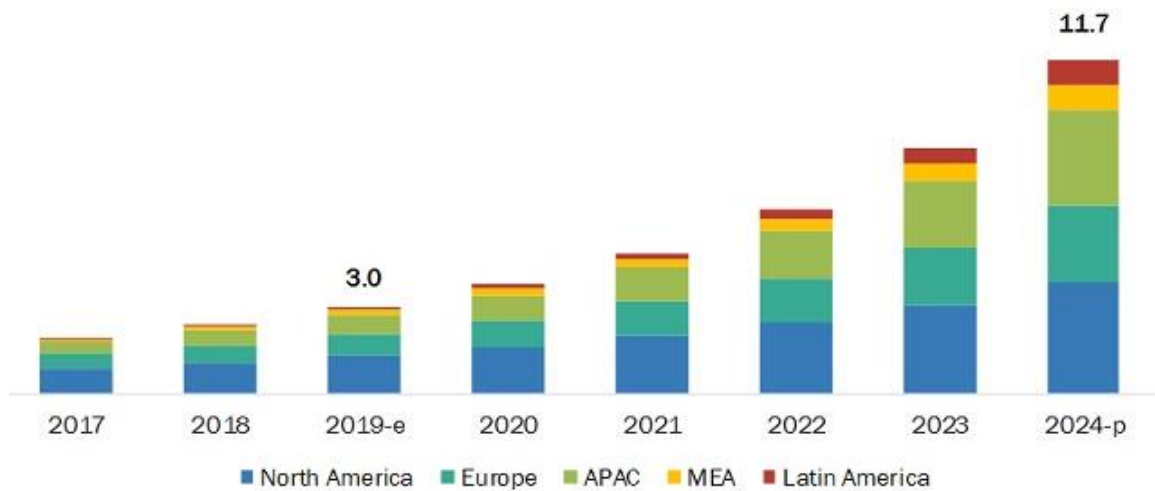


[그림 13] [10]의 악성코드 분류 방법의 개요

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

1.2.2 국내외 시장 현황

세계적인 시장예측 기관인 Markets and Markets의 조사에 따르면 2019년 세계 악성코드 분석 시장 규모는 약 30억 달러(한화 3조 6,270억 원¹⁾) 규모이고, 연평균 성장률(compound annual growth rate, CAGR)은 31.0%로 2024년에는 117억 달러(한화 14조 1,453억 원¹⁾) 규모로 성장할 것으로 예측했다 [13].




[그림 14] 악성코드 분석 시장규모 및 예측치 (단위: USD Billion) [13]

악성코드 분석과 관련된 국내외 시장은 빠르게 성장하고 있고, 딥러닝 기반 악성코드 분석기에 대한 연구가 활발하게 진행되고 있는 한편, 1.2.1 항에서 살펴본 것처럼 우리의 목표인 딥러닝 기반 악성코드 분석 보조도구에 대한 연구는 전무한 상황이다.

[표 1]은 국내외 주요 수요처를 나열한 것이다. 본 기술은 정보보안 분석도구로 제공되기 때문에 정확한 수요량을 파악하기 어려우므로 기재하지 않았다.

¹ 환율 1209원 기준 (2020-06-06 확인)


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

[표 1] 국내외 주요 수요처 현황

수요처	국명
국가정보원	대한민국
군사안보지원사령부(구 국군기무사령부)	대한민국
한국정보통신기술협회(TTA)	대한민국
한국기계전기전자시험연구원(KTC)	대한민국
국가보안기술연구소(NSR)	대한민국
한국인터넷진흥원(KISA)	대한민국
AhnLab	대한민국
SKinfosec	대한민국
EST Security	대한민국
WINS	대한민국
Penta Security	대한민국
IGLOO Security	대한민국
root9B	미국
CheckPoint	이스라엘
Sophos	영국
DarkTrace	영국
Qihoo 360	중국

1.2.3 국내외 경쟁기관 현황

정보보안기업은 새로운 보안위협에 대응하기 위해 신종 악성코드 및 스파이웨어를 분석하고 악성코드 분석 엔진을 업데이트하기 위해 보안 전문가로 이루어진 대응 조직을 운영하고 있다. 국내에서 바이러스 분석 및 대응 소프트웨어로 유명한 V3, 알약을 각각 개발한 AhnLab과 EST security는 이러한 대응 조직으로 각각 안랩 시큐리티대응센터(AhnLab Security Emergency response Center, ASEC) [14], 이스트시큐리티 시큐리티대응센터(ESTsecurity Security Response Center, ESRC) [15]를 운영하고 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 15] 안랩 시큐리티대응센터의 대응 프로세스 [14]


각 대응센터에서는 악성코드 분석 보조도구가 없거나, 공개하지 않는 상황이다. 따라서 보안에 대한 투자가 적은 기업이나 기술력이 충분하지 않은 기업은 빠르고 정확한 악성코드 분석이 어렵다.

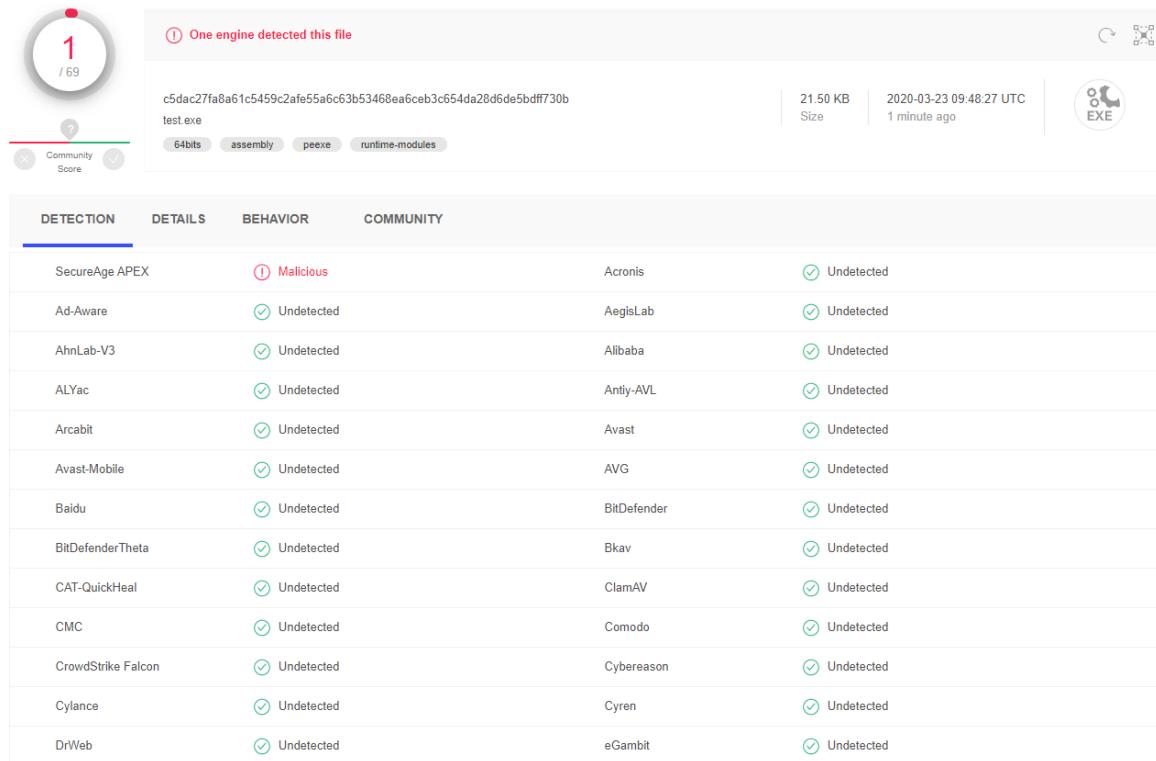
한편, 이후에 살펴볼 바이러스 분석 프로그램 개발기업 현황에서 우리가 제안하는 악성코드 분석 보조도구 프로그램은 전무하고, 악성코드 자동 분석기의 개발만 활발히 진행되고 있는 상황으로 우리가 제안하는 기술의 개발이 시급하다.

● VirusTotal [19]

VirusTotal 은 여러 바이러스 검사 소프트웨어 엔진을 이용해 업로드된 파일 또는 URL(uniform resource locator)의 악성여부를 검사하는 웹사이트이다. 악성코드 검사 엔진으로는 안랩, 알약, Kaspersky, McAfee, 등 72 개의 악성코드 분석 엔진을 사용하며, domain 검사 엔진으로는 Alexa, Google Safebrowsing, 등 68 개의 분석 엔진 및 데이터 셋을 사용한다.

[그림 16]은 C 언어로 작성한 소스코드를 Visual Studio 2019 로 컴파일한 예제 실행 파일을 VirusTotal 에 업로드한 결과이다. 파일 타입 등의 문제로 검사하지 못한 일부 엔진을 제외한 총 69 개의 분석 엔진으로 검사한 결과가 표시되어 있다. 악성 행위를 수행하지 않는 코드임에도 불구하고 SecureAge APEX 엔진에서는 악성으로 오탐한 것을 볼 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06




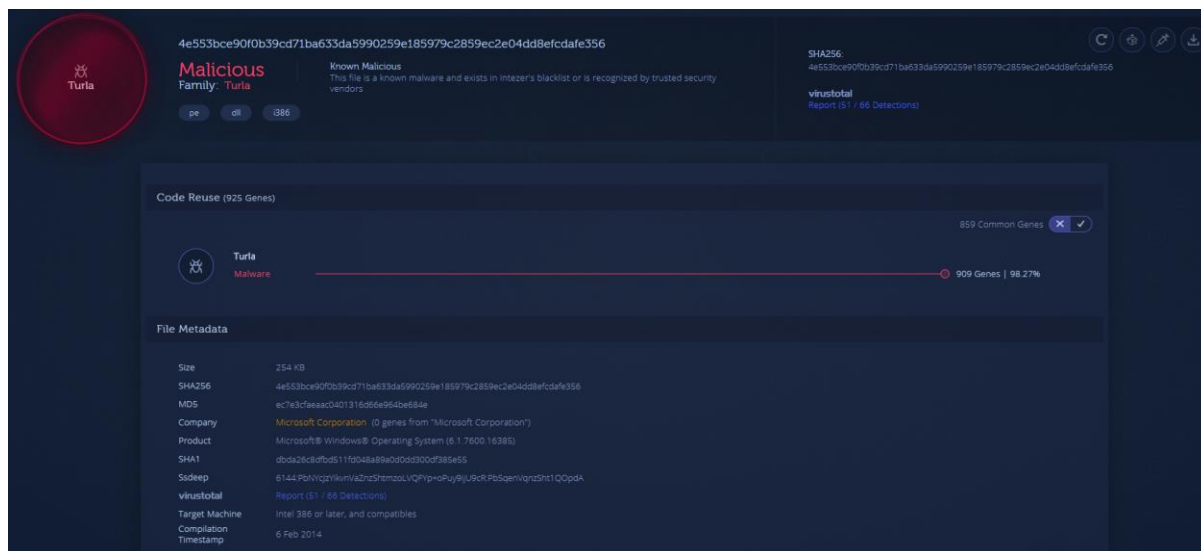
[그림 16] VirusTotal 정상파일 분석 결과

● Intezer [20]

Intezer 는 악성코드 유전자 분석 기법(genetic malware analysis)을 적용해 악성코드 패밀리를 분류하는 회사이다. 악성코드를 작은 조각(유전자)으로 나눈 뒤, 각 작은 조각을 데이터베이스의 코드 조각들과 비교하여 코드의 악성여부를 판별한다.

[그림 17]은 Turla 악성코드를 분석한 결과이다. 분석 결과 총 925 개의 코드 조각 중 909 개가 Turla 와 유사한 것을 볼 수 있으며, 이는 VirusTotal 의 분석 결과 66 개의 악성코드 분석 엔진 중 51 개의 엔진에서 악성여부를 탐지한 것과 일치한다. 분석 결과에서 추가로 파일의 메타데이터(metadata)와 스트링 정보 등을 추가로 확인할 수 있다.

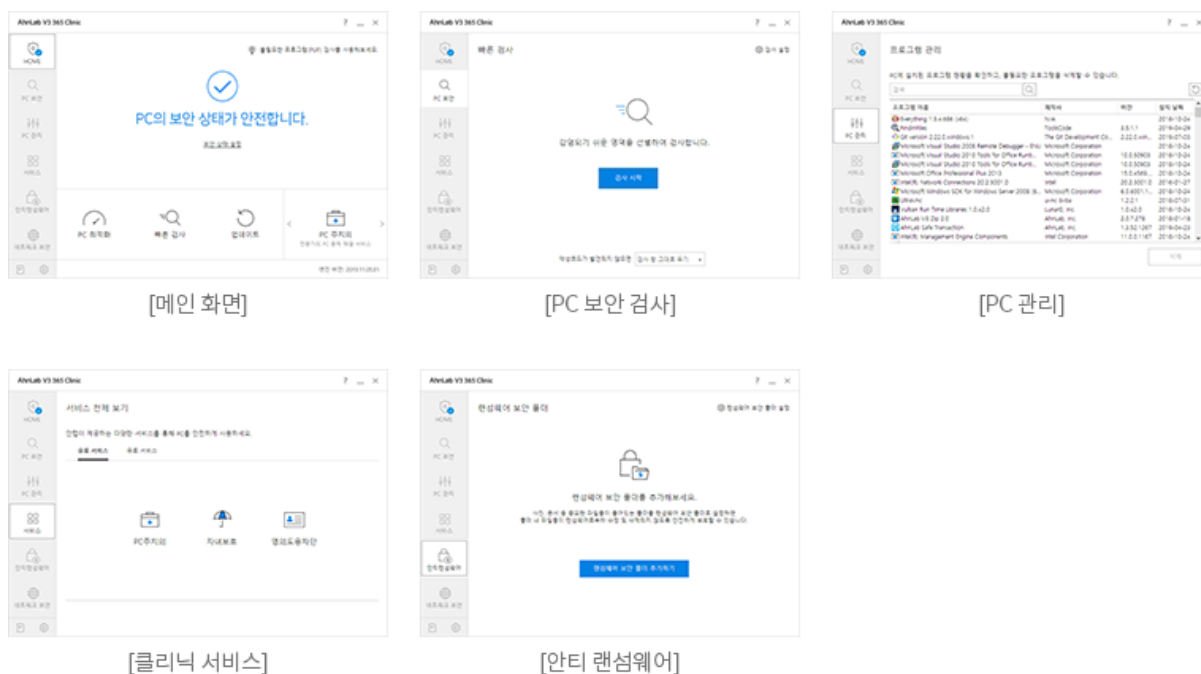
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06




[그림 17] Intezer의 Turla 악성코드 분석 결과

● V3 [21]

V3는 AhnLab에서 개발한 보안 프로그램이다. 시스템에 저장되어 있는 파일을 행위 기반, 평판 기반으로 분석 및 대응하는 것이 주 목적이며, 정상 또는 악성 여부가 불분명한 파일은 클라우드 기반으로 분석하여 변종 악성코드에 대응하는 기능을 제공한다.

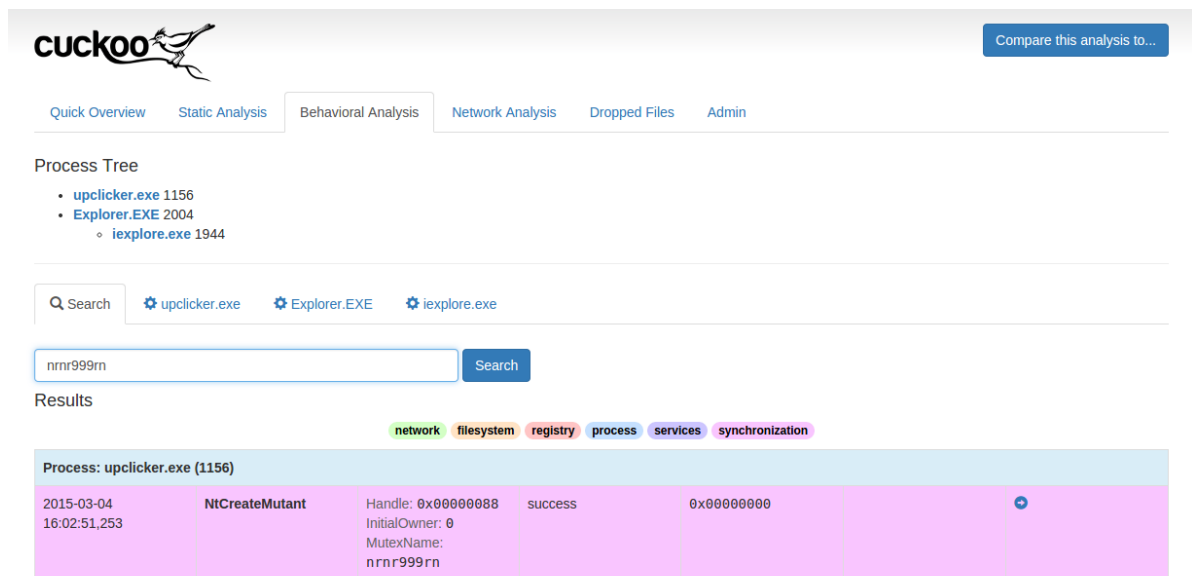


[그림 18] V3 365의 주요 기능

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

● Cuckoo Sandbox [22]

Cuckoo Sandbox 는 악성코드를 자동으로 동적분석하는 시스템으로, 악성코드를 가상환경에서 실행시켜 악성코드의 프로세스, 생성한 파일, 네트워크 트래픽, 메모리, 등을 분석한다. 악성코드를 시스템에서 수행시키면 악영향을 미칠 수 있으므로 독립된 가상환경에서 악성코드를 수행시키고 동작 정보를 수집한다. 분석 가능한 파일 형식은 윈도우 실행파일, DLL 파일, PDF 파일, 등이다.




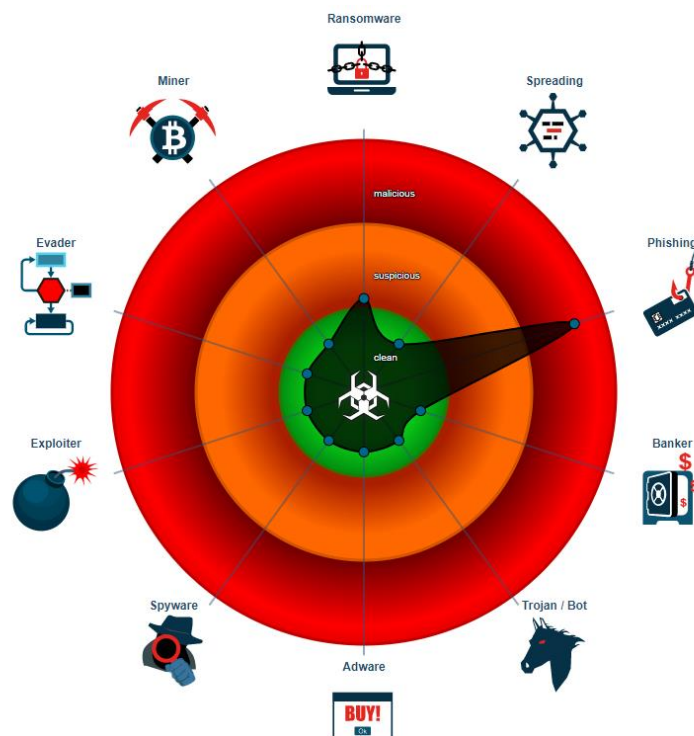
The screenshot shows the Cuckoo Sandbox web interface. At the top, there's a navigation bar with tabs: Quick Overview, Static Analysis, Behavioral Analysis, Network Analysis, Dropped Files, and Admin. Below this, the 'Process Tree' section shows a tree structure: upclicker.exe 1156, Explorer.EXE 2004, and iexplore.exe 1944. A search bar is present with the text 'nnr999rn'. The 'Results' section shows a table of analysis results for the process 'upclicker.exe (1156)'. The table has columns for timestamp, operation, handle, result, and other details. The first row shows a successful 'NtCreateMutant' operation on 2015-03-04 at 16:02:51.253.

[그림 19] Cuckoo sandbox의 동적 분석 결과

● Joe Sandbox [23]

Joe Sandbox 는 악성코드를 자동으로 동적분석하는 시스템으로, 악성코드를 가상환경에서 실행시켰을 때의 동작 정보를 분석한 뒤, [그림 20]과 같이 결과를 시각화하여 제시한다. 이 프로그램은 Windows, macOS, Linux, Android, iOS 와 같은 다양한 플랫폼에서 악성파일, 악성문서가 동작할 때의 분석 결과를 제공하는 장점이 있다.

 <div> <p>국민대학교</p> <p>컴퓨터공학부</p> <p>캡스톤 디자인 I</p> </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 20] Joe Sandbox의 동적 분석 결과

1.2.4 딥러닝 기반 악성코드 분석 보조도구의 필요성


딥러닝 기반의 분류기는 다음과 같이 두 가지 문제점이 있으므로 일부 악성코드에 대해서는 전문가의 추가 분석이 필요하다.

● 정확도 문제

1.2.1 에서 살펴본 어떠한 연구 결과에서도 100%의 정확도를 갖는 악성코드 자동 분석기를 개발하지 못했다. 그러나, 분석기가 하나의 악성코드라도 정상이라 판정해 적절한 대응을 하지 못한다면, 시스템의 보안에 큰 위협이 될 수 있다. 따라서 자동 분석 결과가 불명확한 경우, 즉 파일의 정상/악성 여부를 높은 신뢰도로 판정하지 못하는 경우, 전문가의 추가 분석이 필요하다.


● Zero-day attack

딥러닝의 성능은 데이터셋의 영향을 크게 받는다. 딥러닝 기반의 악성코드 자동 분석기 역시 학습에 사용한 데이터셋에 있는 악성코드나, 일부 변형된 악성코드는 높은 정확도로 판별할 수 있지만, 새롭게 개발된 악성코드를 제대로 분류하지 못하는 문제가 있다. 또한, 악성코드의 변형이 반복되는 경우 이전 악성코드와의 유사도가 떨어져 오래된 데이터셋으

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

로 학습된 분석기가 제대로 분류를 하지 못하는 시간붕괴 문제가 있다. 이 경우 전문가의 추가 분석이 필요하다.

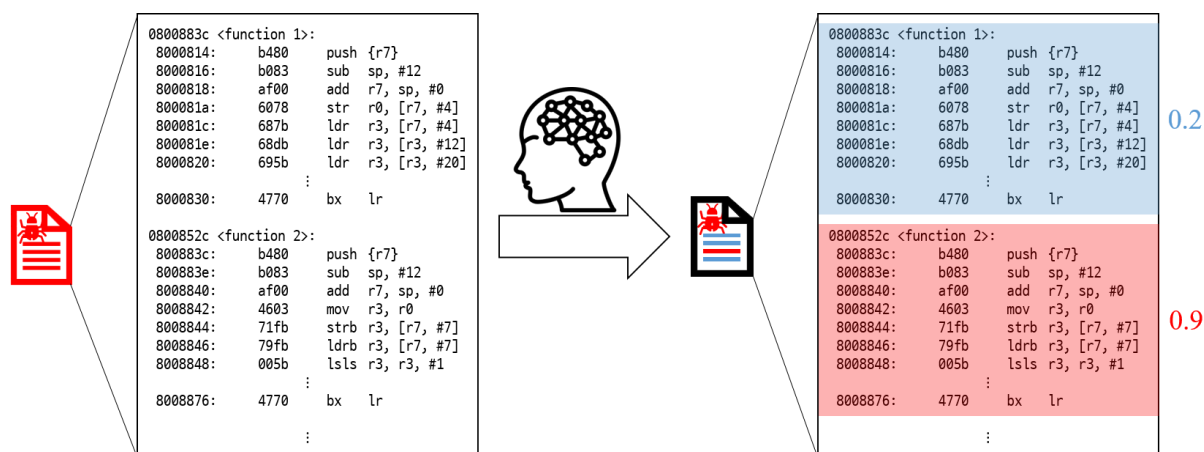
현재 악성코드는 매년 크게 증가하고 있는 반면, 악성코드 분석 인력은 매우 적은 상황이다. 적은 인력으로 높은 분석 효율을 달성해야 하는 현상황에서, 악성코드 분석에 대한 연구는 1.2.1 에서 살펴보았듯 분류기의 성능 향상에 치중되어 있고, 전문가의 분석을 돕는 딥러닝 기반의 보조 도구 연구는 전무하며, 1.2.3 에서 살펴보았듯 악성코드 분석도구 역시 악성행위 판정과 패밀리 분류를 위한 자동분석 도구가 주로 개발되고 있는 상황이다.. 따라서 전문가의 분석 시간을 줄이기 위한 기술 연구가 시급하다

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

2 개발 내용 및 결과물


2.1 목표

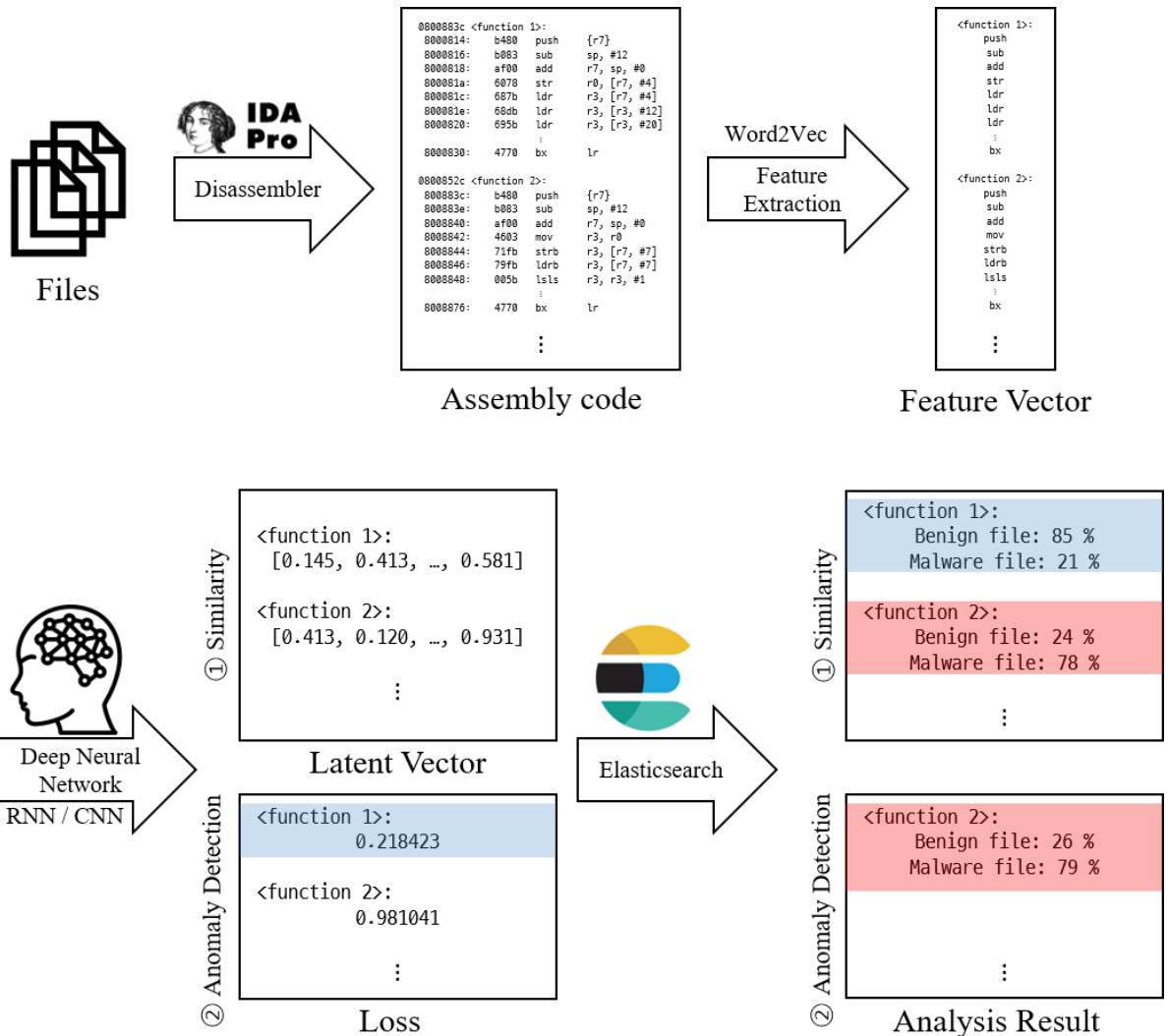
본 프로젝트의 목표는 딥러닝 기반 악성코드 분석 보조도구를 개발하는 것이다. 제안된 프로그램은 파일을 입력으로 받아 함수 단위로 악성 행위 수행 가능성을 점수화 함으로써 전문가의 악성코드 분석을 돕는다. 본 프로그램을 이용하면 적은 인력으로도 악성코드를 신속하게 분석할 수 있으므로 정보보호에 소요되는 비용을 절감하고 신종 악성코드에 신속히 대응할 수 있을 것으로 기대된다.



[그림 21] 기술개발의 최종목표


제안된 소프트웨어의 개요는 [그림 22]와 같다. 우리는 각 함수에 대한 악성행위 수행여부를 점수화하기 위해 두 가지 방법을 도입했다. 첫 번째 방법은 CNN을 기반으로 함수를 잘 나타낼 수 있는 압축벡터(latent vector)를 구하는 알고리즘을 이용하는 것이다. 압축벡터를 계산한 뒤 Elasticsearch [24]를 기반으로 입력 파일의 각 함수와 유사한 함수들을 찾고 해당 함수를 포함하는 파일의 악성/정상 여부를 이용하여 악성 행위 수행 여부를 점수화 한다. 두 번째 방법은 RNN을 기반 오토인코더(autoencoder)로 연산했을 때의 손실값을 이용하는 것이다. 손실값이 특정 임계값(threshold)보다 높다면 해당 함수가 이상 행위를 수행할 것이라 예측하고 Elasticsearch를 기반으로 유사한 함수들을 찾는다. 이후 평가 방안은 첫 번째 방법과 동일한다.

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 22] 제안된 소프트웨어의 분석 방법 개요

소프트웨어 개발의 세부목표는 [표 2]와 같이 크게 세 단계로 나눌 수 있다. 첫 번째 단계는 학습 및 Elasticsearch를 위한 정상코드 및 악성코드 파일 데이터셋을 확보하고, 니모닉(mnemonic) 추출 알고리즘을 개발하는 것이다. 두 번째 단계는 추출한 니모닉을 신경망이 학습할 수 있도록 적절한 특징 벡터 공간으로 변환하는 것이다. 세 번째 단계는 딥러닝 기반 이상탐지를 수행할 수 있는 알고리즘을 확보하고, 이를 기반으로 각 함수에 대한 악성행위 수행 여부를 점수화 하는 단계이다. 또한, 이용자의 컴퓨팅 환경에 관계없이 편리하게 프로그램을 사용할 수 있도록 웹서버를 구축하는 단계이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

[표 2] 기술개발의 세부목표

1 단계 연구 목표	
- 정상코드 및 악성코드 파일 데이터셋 확보	
- 니모닉 추출 알고리즘 개발	
2 단계 연구 목표	
- 딥러닝 기반특징 벡터 추출 도구 개발	
3 단계 연구 목표	
- 딥러닝 기반 이상탐지 알고리즘 확보	
- 딥러닝 기반 자동 악성행위 코드 추출 도구 개발	
- 악성코드 분석 보조도구 제공을 위한 서버 구축	

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용


2.2.1.1. 데이터 수집

학습에 필요한 정상 및 악성파일을 수집하기 위하여 Microsoft의 Kaggle 프로젝트인 Microsoft malware classification의 데이터셋 [16], 한국인터넷진흥원의 정보보호 R&D 데이터 챌린지의 데이터셋 [17]을 활용한다. Microsoft malware classification 프로젝트는 악성파일의 패밀리를 분류하는 것을 목표로 한다. 데이터셋은 바이트 파일 10,868개와 어셈블리 파일 10,868개로 총 21,738개이며, 각 악성파일의 패밀리가 제공된다. 한국인터넷진흥원 정보보호 R&D 데이터 챌린지는 2017년부터 시행된 경진대회이며 정상파일과 악성파일을 분류하는 것을 목표로 한다. 데이터셋은 정상파일 10,000개와 악성파일 10,000개이며, 각 파일의 정상/악성 여부가 제공된다.

본 프로젝트에서는 학습을 위해 정상파일을 사용하므로 양질의 정상파일을 확보해야 한다. 이를 위해 우리는 정상파일을 수집하기 위한 크롤러를 제작했다. 크롤링 대상은 높은 신뢰도로 정상이라 판정할 수 있는 시스템의 DLL(dynamic link library) 파일과 Steam 사의 게임 인스톨러로 지정했다.

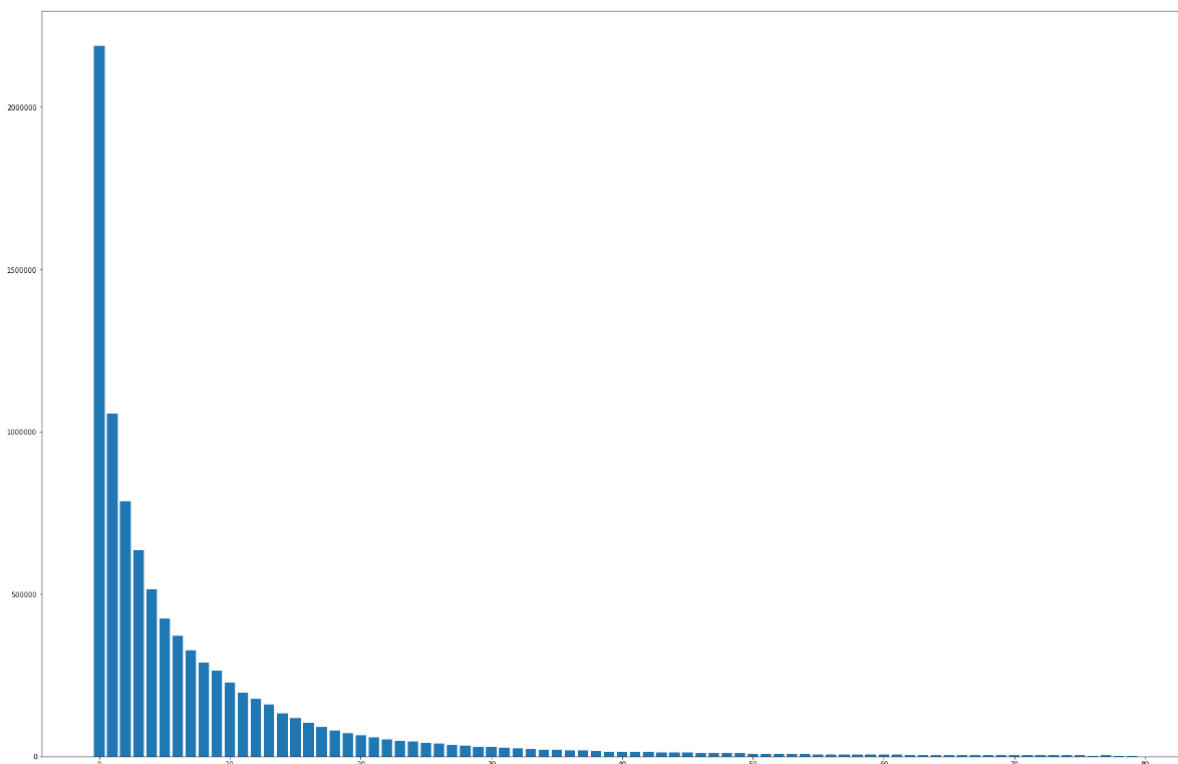
2.2.1.2. 데이터 전처리

수집한 데이터셋은 실행파일과 DLL 파일이므로, 학습을 위해서는 파일 중 어셈블리 코드를 추출하는 과정이 필요하다. 우리는 이를 위해 상용 역어셈블러 프로그램인 IDA Pro [25]를 사용했으

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

며, 역어셈블된 결과 중 니모닉을 추출하기 위해 파서를 Python으로 직접 구현했다. 구현한 파서는 IDA Pro 상에서 동작하며, 역어셈블된 결과를 기본 블록(basic block) 단위로 나눈 것을 Python 피클(pickle)로 저장하도록 구현했다.

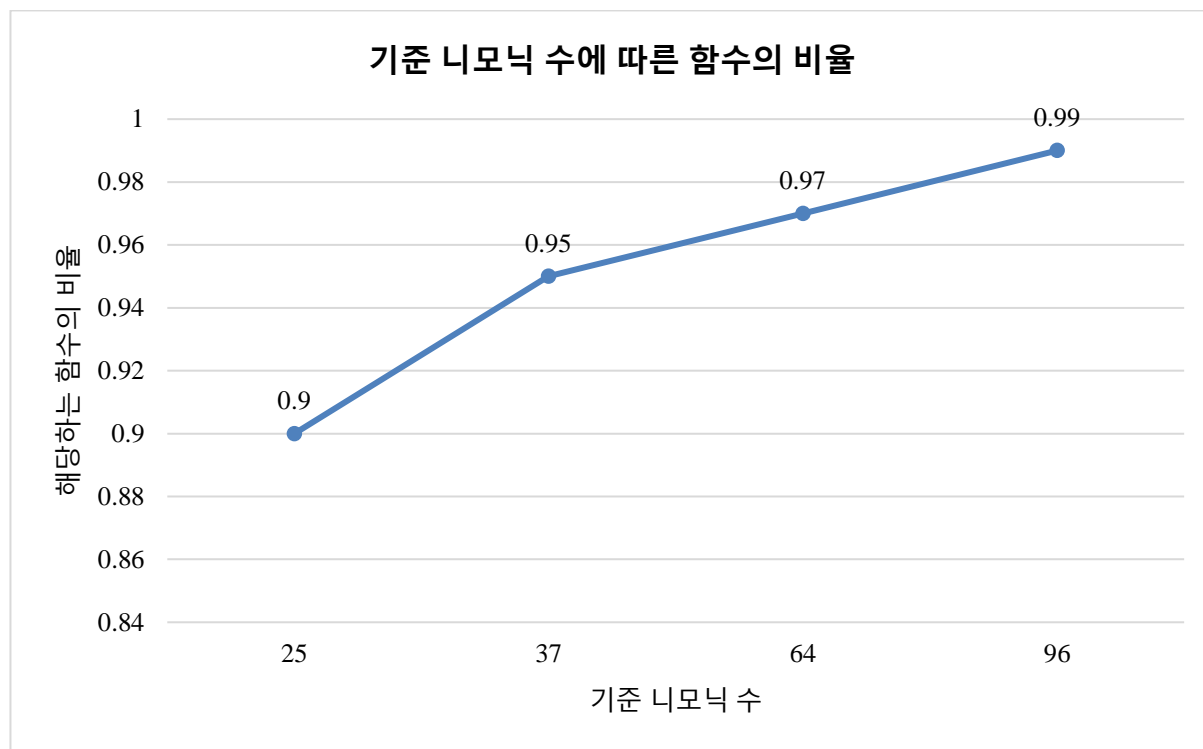
신경망의 입력 벡터의 크기는 고정되어 있으므로, 입력 벡터의 크기를 동일하게 구성하는 전처리가 필요하다. 즉, 분석할 함수의 니모닉의 개수를 일정하게 맞추는 작업이 필요하다. 이때 니모닉의 개수가 너무 적으면 정보를 잃게 되며, 개수가 너무 많으면 신경망의 너비가 넓어져 과적합이 발생할 수 있으므로 적합한 기준 니모닉 수를 찾아야 한다. 적합한 기준을 찾기 위해 우리는 5,000개의 정상 파일의 각 함수 별 연산자의 수를 분석했다. [그림 23]은 이를 도식화한 것으로, x축은 연산자의 수, y축은 함수의 수를 의미한다. 분석 결과 짧은 함수가 대부분의 비중을 차지하며, 니모닉의 수가 증가할수록 해당하는 함수의 수는 감소함을 볼 수 있다.



[그림 23] 각 함수별 니모닉의 수

이를 기반으로 적합한 니모닉의 수를 구하기 위해 100,000개의 정상 파일에 대해 기준 니모닉 수에 따른 함수의 비율을 나타낸 것이 [그림 24]와 같다. 그림에서 니모닉의 수가 64개 이하인 함수의 비율은 97%로 대부분의 함수를 포함하므로 기준 니모닉 수는 64로 지정했다. 입력 함수의 연산자의 수가 기준 니모닉 수보다 작다면, 남은 부분을 “padding”이라는 단어로 패딩하였으며, 이보다 길다면 앞 64개의 연산자만 활용하도록 구성하였다. 또한, 니모닉의 종류가 너무 많아져 학

습에 방해가 되는 것을 방지하기 위해 파생되는 니모닉을 하나의 명령어로 통합하였다. 예를 들어 mov.ovf는 mov로, push.ovf는 push로 통합하였다.



[그림 24] 니모닉 수에 따른 함수의 비율


2.2.1.3. 딥러닝 기반 특징벡터 추출 도구

가) [실험 결과 1] 적합한 임베딩 알고리즘 탐색

우리는 gensim 라이브러리[26]를 이용하여 이상의 세 알고리즘을 구현했다. 하이퍼 파라미터는 디폴트 파라미터를 참고하여 윈도우 크기 10, 최소 단어 수 50, 에포크 10, 학습률 0.0002로 지정했다. RNN 구현시 모델의 너비를 적게 유지하면서 단어를 충분히 적합한 특징 공간으로 사상시키기 위해 우리는 특징벡터의 차원을 8, 16, 32, 64, 128로 각 단어의 의미를 충분히 나타내는 최소 차원을 찾는 것을 목적으로 실험했다. 학습 데이터는 정상 파일 5,000개를 활용했다.

단어가 적합한 특징 공간으로 사상 되었는지 확인하기 위해 우리는 다섯 개의 연산자 mov, jmp, add, pop, push에 대해 가장 가까운 위치에 사상된 연산자를 확인했으며, 사상시킨 결과를 t-SNE로 시각화 했다.


실험 결과 Word2Vec 알고리즘이 임베딩에 가장 적합함을 확인했다. Word2Vec는 시각화 결과 유사한 동작을 수행하는 연산자끼리 가까운 특징 공간으로 임베딩되는 것을 확인했다. [표 3]은 각 특징벡터의 크기별 가까운 니모닉을 나타낸 것이며, [그림 25]는 t-SNE로 임베딩된 니모닉을 시각

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

화한 결과이다. 나머지 실험 결과는 부록에 정리하였다.

[표 3] Word2Vec(Skip-gram)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
mov	lea setns xor test cmovnz cmovz push sets setz cmovs	lea xor cmovnz push cmovb shlx xchg cmovz add cmp	lea push xor xchg cmovnz cmovge cmovz inc sar movsx	lea push inc xchg setnz cmovge sar xor setnl cwde	lea push inc sar xor setnz cmovge imul xchg cmovg
jmp	jge jl jnz jnb jbe lea jb jg mov ja	jl jb jge jnz jg ja retn jle jz jnb	retn jl jge jg jb jle jnz jz jns jbe	retn jg lea jle jnz jl jge js ja jz	retn jl jle jns jnz jge js jb jbe jz
add	pextrb dec prefetcht1 phaddsw cmovz cmp stmxcscr psrldq movd movq	sub shl prefetcht0 mov movups dec shlx adc inc cdq	sub shl prefetcht0 prefetchnta mov movaps sar pshufd cdq movups	sub shl imul sar mov lea neg subpd pshufd cmp	sub shl movaps mov imul lea jg movups sar neg
pop	leave xor fninit vzeroupper fild12e cmpsd retn sfence fcmovb fcmovnb	retn leave vzeroupper xor sfence pushf popf cmpltd cmpltd cvtpd2pi	retn vzeroupper xor emms setz sfence leave setnl setnl mov fninit	retn setz emms mov vzeroupper leave setns setnl setnz push	retn vzeroupper leave mov setz setnl setns jmp setnz push
push	jns lea setns js mov fnclex test stosd cmovs crc32	lea mov test fnclex jmp setns js fld1 fldz cmovns	mov lea cmovs cmovnz cmovge cmovns test cmovl fnclex movsx	lea mov stosd setnz setz cwde test fstp cmovnz cmovz	mov lea test stosd setz setnz fstp cmovnz fldz inc

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 25] Word2Vec(Skip-gram)의 벡터 크기가 32일 때 t-SNE로 시각화한 결과

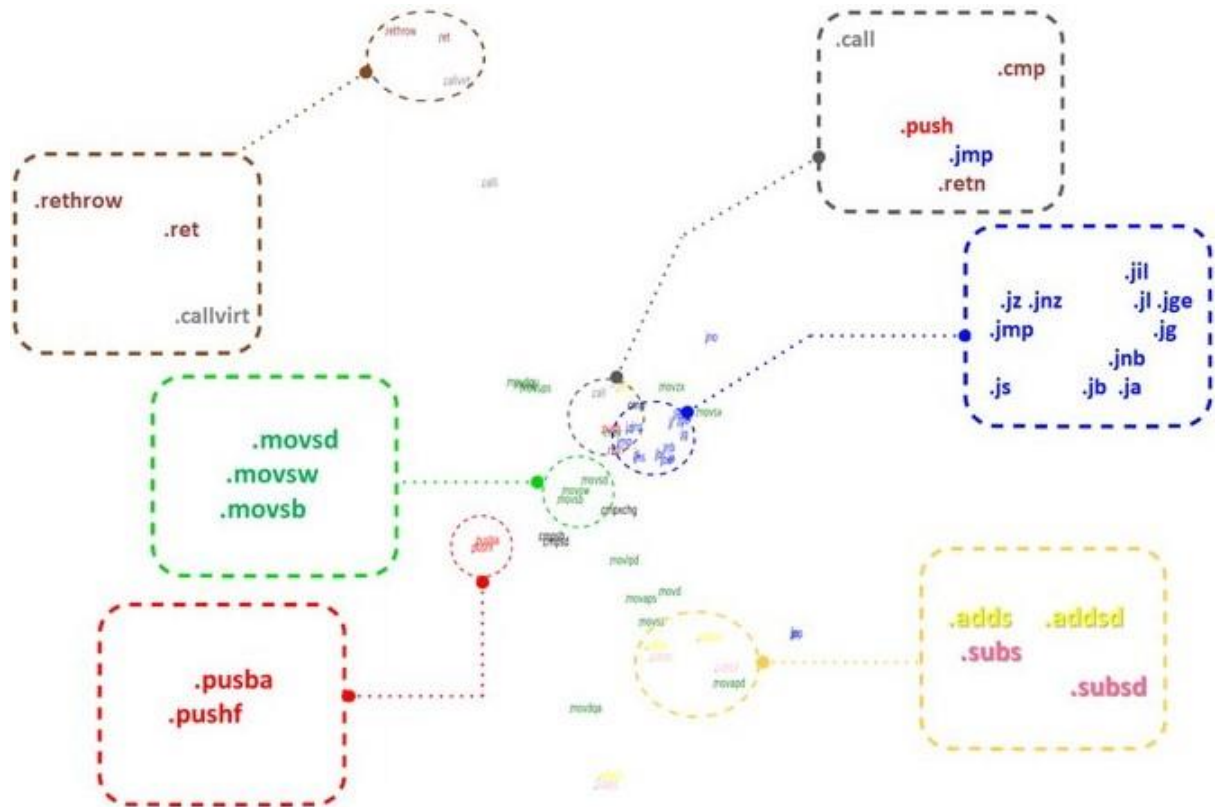
나) [실험 결과 2] 적합한 하이퍼 파라미터 탐색

우리는 Word2Vec 임베딩 알고리즘을 기반으로 적합한 하이퍼 파라미터를 탐색했다. 효과적으로 학습을 수행하기 위해 데이터의 수를 55,000개로 늘렸으며, skip-gram과 CBOW 두 가지 방법을 모두 실험했다.

탐색할 하이퍼 파라미터는 특징벡터의 차원과 윈도우 크기이다. 특징벡터의 차원이 32, 64, 128, 256인 경우와 윈도우 크기가 2, 3, 4인 경우를 탐색한다. 임베딩 알고리즘 두 가지에 대해 실험을 진행하므로 총 24가지 경우에 대한 실험을 수행하였다. 나머지 하이퍼 파라미터는 최소 단어 수 1,000, 에포크 10, 학습률 0.0002로 지정했다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

실험 결과는 [그림 26]과 같다. Word2Vec(CBOW) 임베딩 알고리즘을 사용하여, 윈도우 크기는 2, 특징 벡터의 차원을 64로 하여 학습을 수행했을 때, 그림과 같이 유사한 동작을 수행하는 명령어끼리 가까운 위치로 임베딩 되었음을 확인했다.



[그림 26] Word2Vec(CBOW) 학습 결과

2.2.1.4. 신경망 실험 결과

본 목에서는 2.1절에서 설명한 두 가지 분석 방안을 실험한 결과를 설명한다. 계획서상에서는 RNN 기반 명령어별 이상탐지 방안을 수행할 예정이었으므로 해당 내용에 대한 실험 결과도 설명한다. 그러나, 실험 결과 명령어별 이상탐지보다 오토인코더기반 이상탐지가 더 효과적이었으므로 프로젝트 결과물에서는 상술한 두 가지 분석 방안만 사용했다. 실험에서 신경망은 Keras 라이브러리[27]를 이용해 구현하였다.

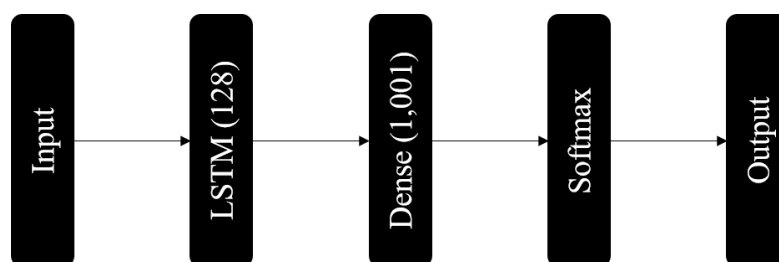
가) 명령어별 이상탐지

주변 명령어로 하나의 명령어를 예측해 각 명령어의 이상 여부를 판정하는 방법은 다음과 같다.

1. 특징벡터를 임베딩 시킨다.
2. 특정 윈도우에 있는 명령어들을 입력 받아 다음 명령어, 혹은 중심 명령어를 예측한다.

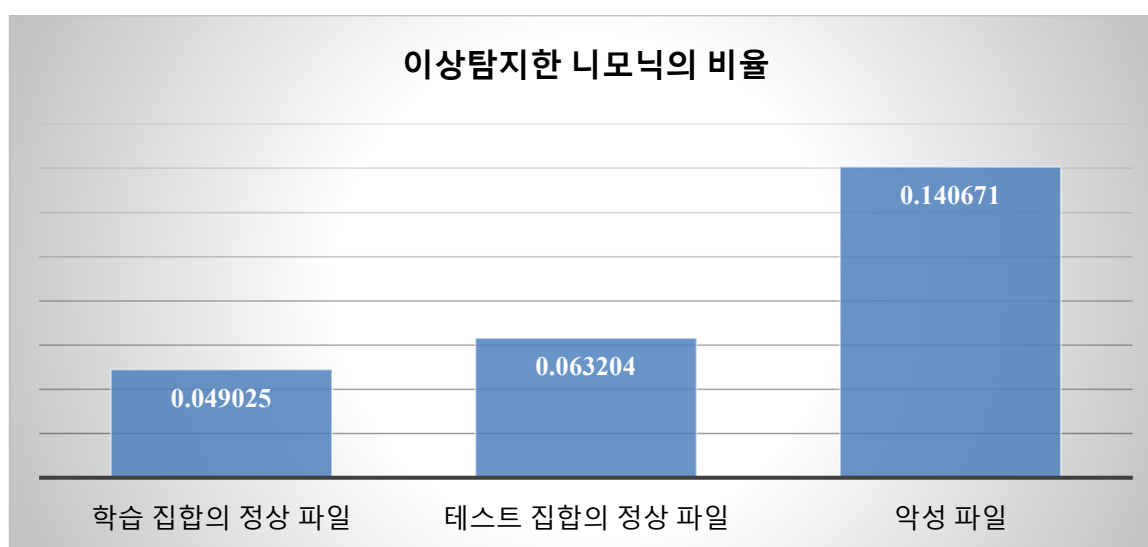
3. 신경망이 예측한 명령어 중 상위 90% 이내에 포함되지 않으면 이상 명령어라 판정한다.

우리는 이전 명령어들로 다음 명령어를 예측하는 방식으로 실험을 수행했다. 하이퍼 파라미터는 다음과 같다. 단어 집합의 개수는 니모닉 1,000개와 알 수 없는 니모닉을 포함하여 1,001 개로 지정했으며, 윈도우 크기는 30으로 지정했다. 신경망은 [그림 27]과 같이 LSTM 한 층에 완전 연결층(fully-connected layer, dense layer)을 이은 신경망을 활용하였다.



[그림 27] 명령어별 이상탐지를 위한 신경망의 구조

1,000개의 파일에 대해 학습을 수행하고, 학습 집합의 정상파일, 테스트 집합의 정상파일, 악성 파일에 대해 이상탐지를 수행한 결과가 [그림 28]과 같다. 그림에서 악성파일에 대한 이상탐지 비율은 정상파일에 대한 이상탐지 비율의 두 배 이상임을 볼 수 있다. 그러나, 이 방법은 니모닉을 하나씩 예측하기 때문에 학습을 위한 시간이 많이 소요되고, 고사양의 컴퓨팅 환경을 필요로 하므로 실질적인 활용이 어렵다. 우리는 이를 개선하기 위해 나)에서 설명할 여러 니모닉을 한 번에 추출하는 오토인코더 기반 이상탐지 방법론을 적용할 예정이다.



[그림 28] 테스트 집합별 이상탐지한 니모닉의 비율

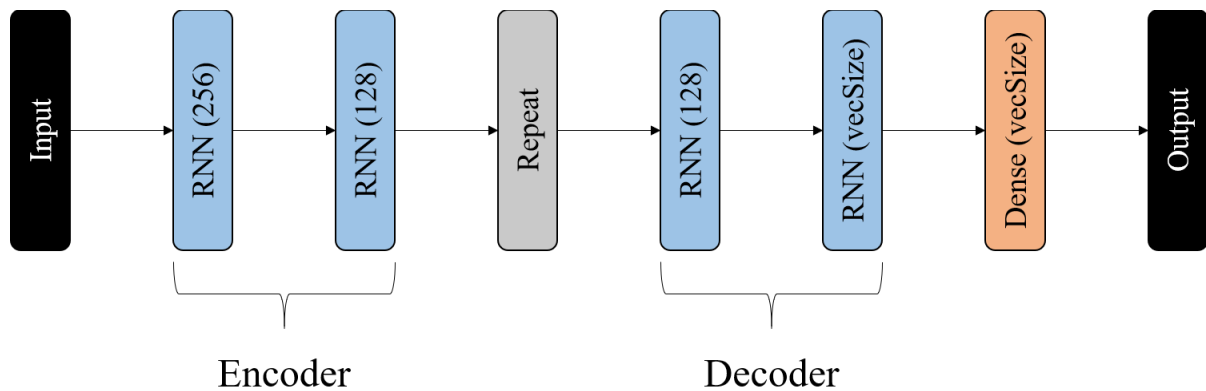
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

나) 오토인코더 기반 이상탐지

오토인코더를 이용해 각 함수별 이상탐지를 수행하는 방법은 다음과 같다.

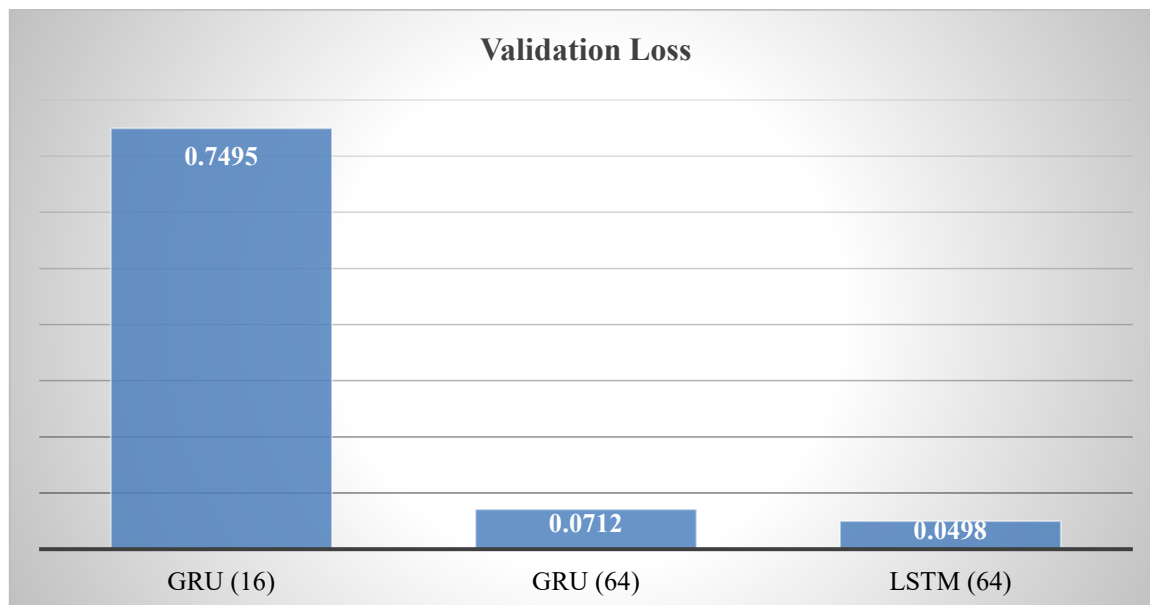
1. 특징벡터를 임베딩 시킨다.
2. 각 함수별 명령어를 입력 받아 해당 함수의 명령어를 예측한다.
3. 신경망이 예측한 명령어와 원본 명령어의 MAE(mean absolute error)를 계산하여, MAE가 특정 임계값보다 크면 이상행위를 수행하는 함수라 판정한다.

실험한 오토인코더의 구조는 [그림 29]와 같다. 그림에서 RNN으로는 GRU와 LSTM을 사용했으며 임베딩 벡터의 크기(vecSize)는 16, 64로 지정해 실험했다. 학습 데이터는 정상파일 10,000개를 사용했다.



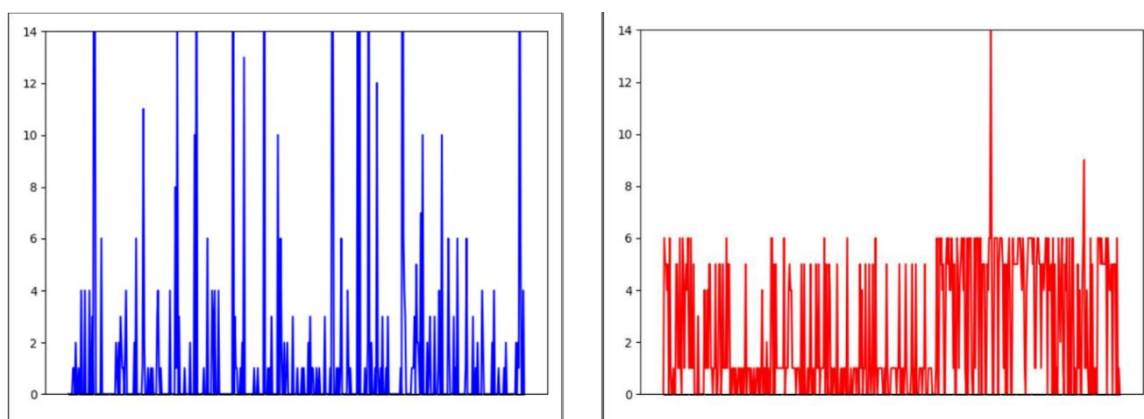
[그림 29] 오토인코더 기반 이상탐지를 위한 신경망의 구조

신경망 구조별 학습 결과가 [그림 30]과 같다. 그림에서 x축은 사용한 RNN 구조와 특징벡터의 차원을 나타낸다. 학습 결과 GRU (16)은 검증 집합의 손실값이 큰 반면, GRU (64)와 LSTM (64)는 낮다. 따라서 실험 결과는 2.2.1.3항의 결과와 같이 특징벡터의 차원은 최소 32차원 이상이어야 함을 시사한다.



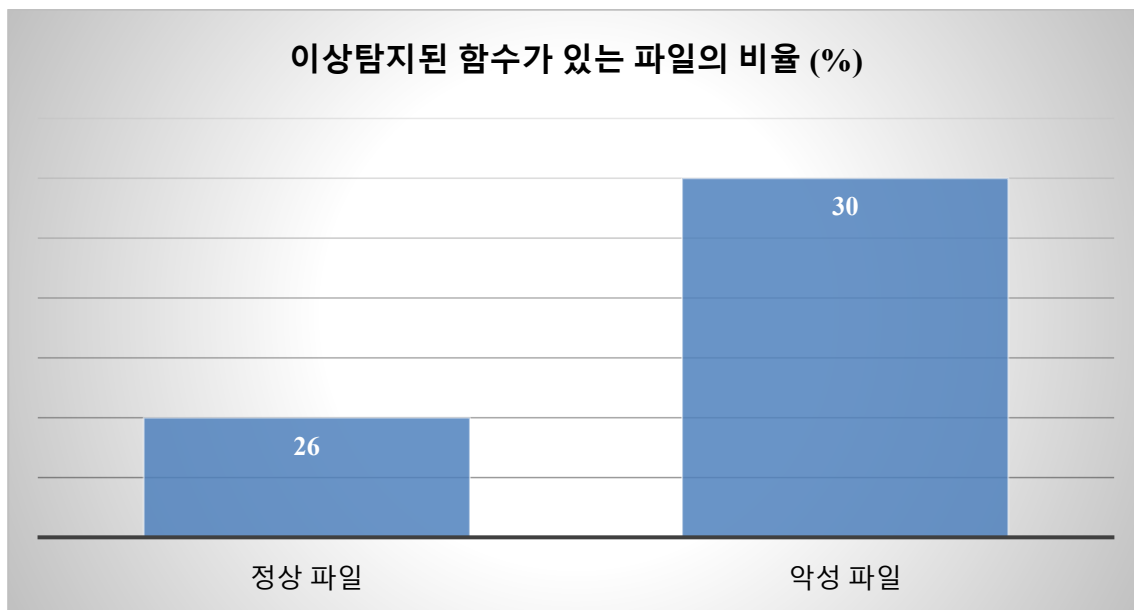
[그림 30] 신경망의 구조별 검증 집합에 대한 손실값

신경망이 학습을 제대로 수행했는지 확인하기 위해 정상파일 500개와 악성파일 500개에 대해 이상 탐지된 함수의 개수를 확인하였다. 이때 임계값은 임의로 0.1이라 지정하였다. [그림 31]은 각 파일 별 이상 탐지된 함수의 개수를 도식화한 것이다. 그림에서 정상파일은 이상탐지가 되지 않거나, 10개 이상의 많은 함수에 대해 이상탐지가 되는 반면, 악성파일은 대부분 6개의 함수에 대해 이상탐지가 되는 것을 볼 수 있다. 특정 정상파일의 함수들에 대해 이상탐지가 되는 이유는 학습에 충분히 많은 데이터를 사용하지 않았기 때문으로 추정된다. 한편, 모든 함수에 대해 이상탐지가 되지 않은 파일의 비율은 [그림 32]와 같다. 그림에서 정상파일은 전체 500개의 파일 중 약 26%가 이상탐지된 함수가 있는 반면, 악성파일은 약 70%가 이상탐지된 함수가 있었다.



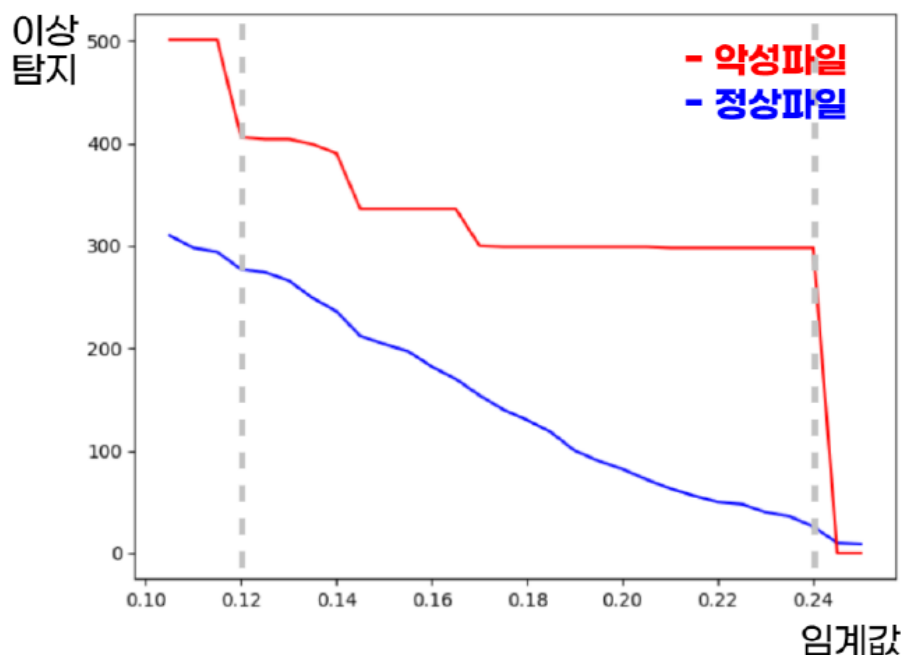
[그림 31] 정상파일과 악성파일의 이상탐지된 함수의 개수

(x축: 파일, y축: 이상탐지한 함수의 개수), (왼쪽: 정상파일, 오른쪽: 악성파일)



[그림 32] 이상탐지된 함수가 있는 파일의 비율

본 이상탐지 방안을 적합하게 수행하기 위해서는 적합한 임계값을 지정하는 것이 필요하다. 이를 위해 우리는 여러 임계값에 대해 정상파일과 악성파일이 이상탐지된 비율을 탐색하였다. [그림 33]에서 볼 수 있듯 임계값이 너무 작으면 오탐, 즉 악성행위를 수행하지 않음에도 악성행위를 수행한다 판정하는 비율이 크고, 임계값이 너무 크면 미탐, 즉 악성행위를 수행함에도 악성행위를 수행하지 않는다고 판정하는 비율이 컸다. 특히 임계값이 0.12보다 작으면 오탐하는 비율이 크고, 임계값이 0.24보다 크면 미탐하는 비율이 크므로, 우리는 아래의 그림을 토대로 적합한 임계값이 0.2라 판정하였다.



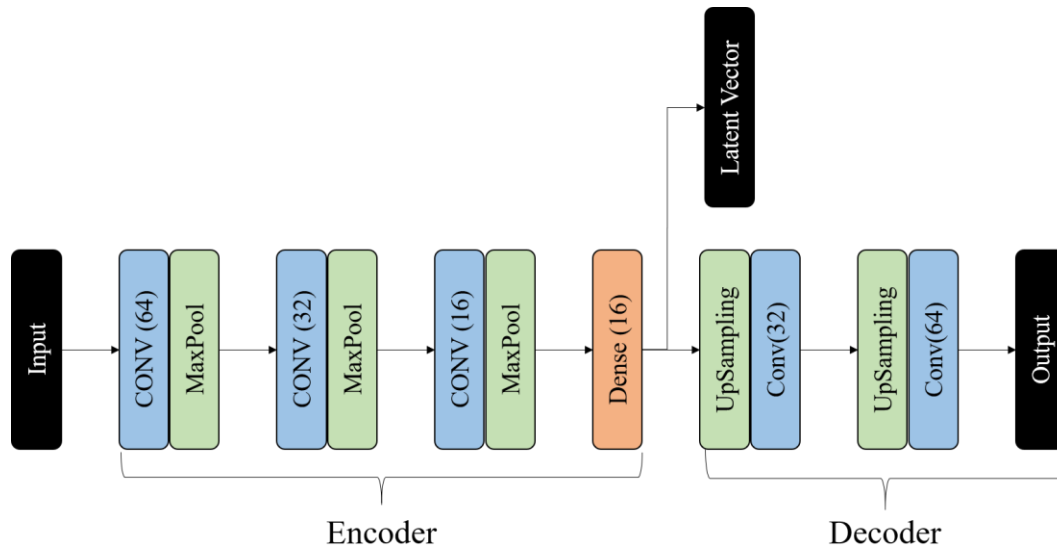
[그림 33] 임계값에 따른 정상파일과 악성파일의 이상탐지된 정상 함수의 수

다) 압축벡터 기반 이상탐지

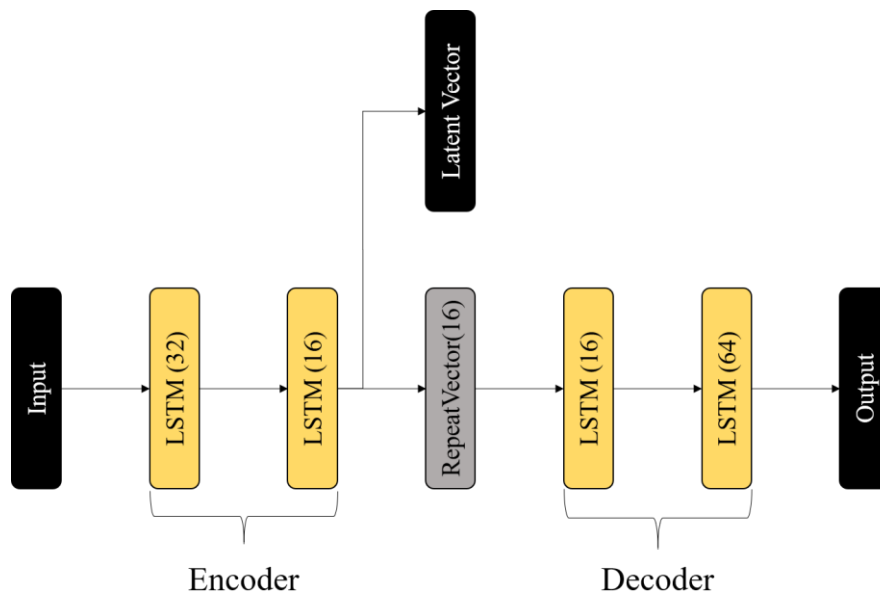
제안서의 방법은 실제 데이터에 대해 만족할 만한 성능을 보이지 못했다. 우리는 이를 보완하기 위하여 함수의 압축벡터(latent vector)를 추출하는 신경망을 구성하고, Elasticsearch를 기반으로 입력 함수와 유사한 함수를 찾아 함수의 악성 행위 가능성을 점수화 하는 방법을 고안했다. 자세한 분석 방법은 다음과 같다.

1. 신경망을 이용해 압축벡터를 구한다.
2. Elasticsearch로 데이터베이스에서 압축벡터와 코사인 유사도가 높은 함수를 찾는다.
3. 찾은 함수 중 악성 행위를 수행하는 함수의 비율로 입력 함수의 점수를 책정한다.

신경망은 [그림 34], [그림 35]와 같이 CNN 기반 오토인코더와 LSTM 기반 오토인코더를 구현하였으며 인코딩 결과를 압축벡터로 사용하였다. CNN 기반 오토인코더에서 컨볼루션층의 커널의 크기는 3, 풀링층의 커널의 크기는 2로 지정했으며, 업샘플링 크기도 풀링층의 커널의 크기와 동일하게 2로 지정했다. 두 신경망 모두 압축벡터와 출력을 계산하는 층에서 활성화함수는 sigmoid를 사용했으며, 이외의 층에서는 ReLU를 사용했다.




[그림 34] CNN 기반 오토인코더 구조



[그림 35] LSTM 기반 오토인코더의 구조

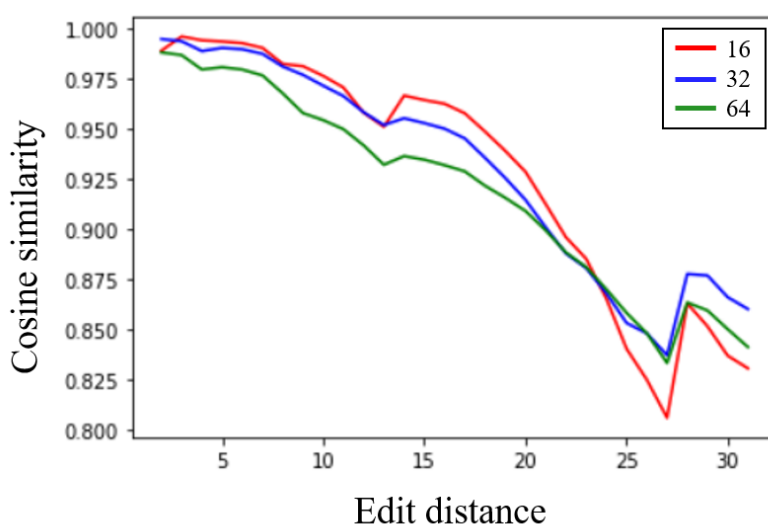
손실함수는 binary cross entropy, 최적화함수는 Adam으로, 이외의 하이퍼 파라미터는 디폴트 파라미터로 하여 CNN은 16에포크, LSTM은 8에포크로 5,147,572개의 함수를 학습시킨 결과 마지막 손실값은 각각 0.6652, 0.6667이었다.

제안한 방법의 적합성을 검증하기 위해 압축벡터의 유사성이 실제 두 함수의 유사성을 의미하는지 확인하였다. 우리는 두 함수의 유사성을 판정하는 기준으로 편집거리(edit distance)를 사용하였다. 편집거리는 두 문자열 A, B에 대하여 A를 B로 변경하기 위해 수정해야 하는 횟수를 의미하는 것으로 편집거리가 낮음은 A와 B가 유사함을 의미한다. 압축벡터의 유사성은 코사인 유사도

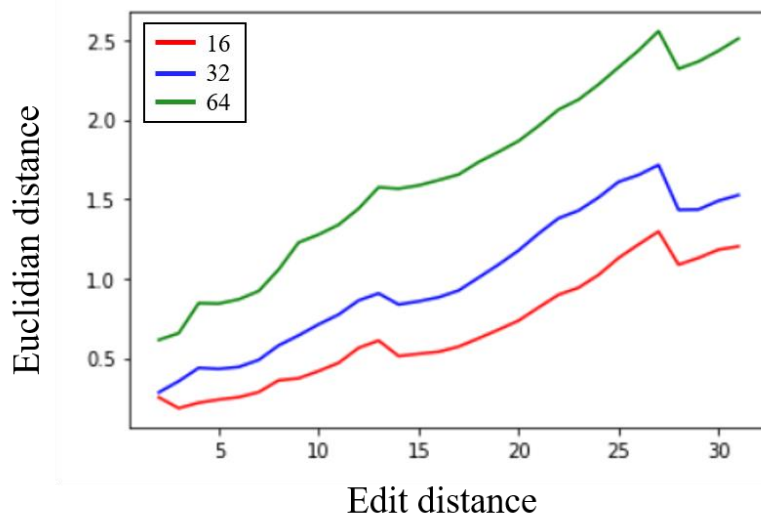
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

(cosine similarity)와 유클리드 거리(Euclidian distance)를 기준으로 판정하였다. 코사인 유사도는 두 벡터가 이루는 각에 대한 코사인 값을 계산한 것으로, 코사인 유사도가 높을수록 두 벡터는 유사함을 의미한다. 유클리드 거리는 두 벡터의 거리를 유클리드 기하에서 측정한 것으로 유클리드 거리가 낮을수록 두 벡터는 유사함을 의미한다. 따라서 제안한 방법이 적합하다면, 편집거리와 코사인 유사도는 반비례해야 하며, 편집거리와 유클리드 거리는 비례해야 한다.

[그림 36], [그림 37]은 각각 CNN 기반 인코딩 결과로 나온 압축벡터에 대한 코사인 유사도, 유클리드 거리와 해당 함수의 편집거리의 관계를 도식화한 것이다. 그림에서 편집거리와 코사인 유사도는 반비례 관계에 있으며 편집거리와 유클리드 거리는 비례 관계에 있으므로 제안한 방법은 적합한 압축벡터를 생성한다. 또한, 우리는 압축벡터의 차원의 증가가 정확도에 영향을 주는지 확인하기 위해 차원을 16, 32, 64로 변환하여 실험을 수행하였다. 실험 결과 압축벡터의 차원은 정확도에 크게 영향을 주지 않음을 확인하였다.



[그림 36] CNN 기반 인코딩 결과 편집거리와 코사인 유사도의 관계



[그림 37] CNN 기반 인코딩 결과 편집거리와 유클리드 거리의 관계

Elasticsearch를 기반으로 유사한 함수를 검색한 결과가 [그림 38]과 같다. 그림에서 왼쪽은 검색을 수행할 함수이고, 오른쪽은 수행 결과 가장 높은 유사도로 나온 함수이다. 실험 결과 유사한 명령어를 갖는 함수가 검색된 것을 볼 수 있다. 이는 압축벡터의 유사성과 함수의 유사성이 일치함을 의미한다.

000091e9cc8946301647fbd01fed6ce1

Index : 4814


```
ldarg.0
callvirt
callvirt
call
stloc.0
ldarg.1
callvirt
callvirt
call
stloc.1
ldloc.0
ldnull
call
brfalse
ldloc.1
ldnull
call
brfalse
ldarg.0
ldarg.1
callvirt
ldloc.1
ldloc.0
call
ret
```

83b24d182dec262ce606c4ab46894c59

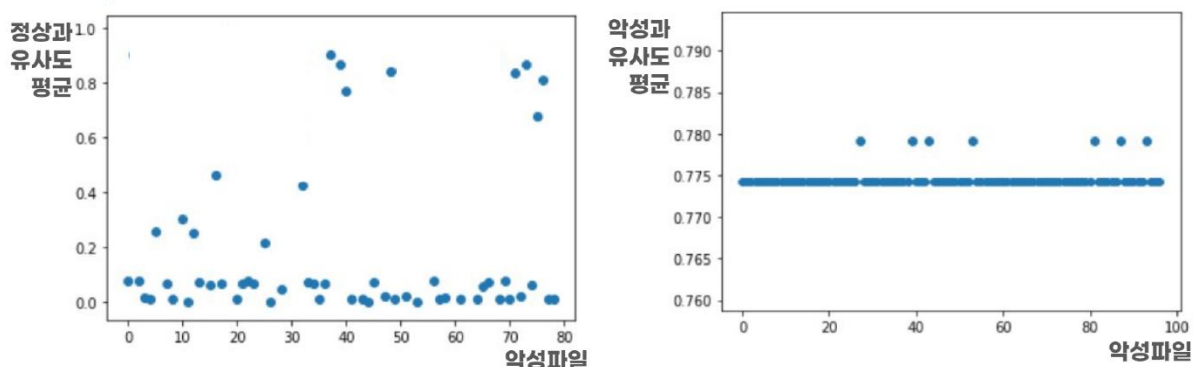
Index : 10977

```
ldarg.0
call
callvirt
ldarg.1
ldfld
callvirt
stloc.0
ldloc.0
brtrue
ret
ldarg.1
ldfld
ldarg.0
ldftn
newobj
callvirt
ret
```

[그림 38] 왼쪽 함수에 대한 Elasticsearch 결과 가장 유사한 함수 (오른쪽 함수)

	국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서	
		프로젝트 명	asi(a security insight)
		팀 명	assist(a security safety important special team)
		Confidential Restricted	Version 1.0 2020-JUN-06

제안한 결과가 적합함을 판정하기 위해 악성 파일을 입력으로 하여 CNN 기반 압축벡터를 계산하고 정상파일과의 유사도의 평균, 악성파일과의 유사도의 평균을 구한 것이 [그림 39]와 같다. 그림에서 악성파일과 정상파일들의 유사도의 평균은 대부분 0.07 정도로 낮고 일부 파일들에 대해서만 높은 유사도를 갖는다. 반면, 악성파일과 악성파일들의 유사도의 평균은 대부분 0.78 정도로 높음을 볼 수 있다. 이는 제안한 방법이 정상파일 및 악성파일과의 유사성을 판정하는 기준으로 적합함을 시사한다.




[그림 39] 제안한 방법을 기반으로 악성파일의 유사도를 구한 결과
(왼쪽: 악성파일과 정상파일들의 유사도의 평균, 오른쪽: 악성파일과 악성파일들의 유사도의 평균)

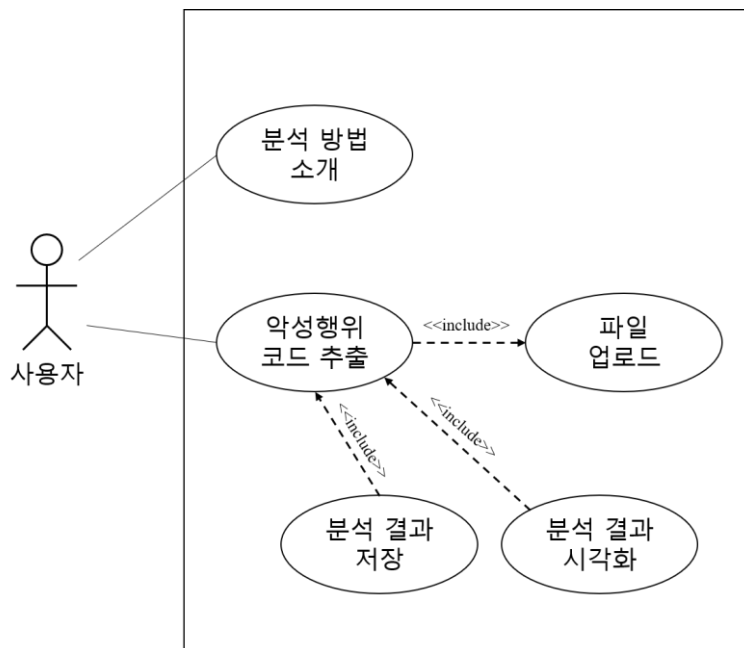
2.2.1.5. 웹 서비스

웹 결과 그림과 함께 상세 내용 추가 예정

2.2.2 시스템 기능 요구사항

계획서의 시스템 기능 요구사항은 [그림 40]과 같으며, 본 소프트웨어는 계획서의 기능 요구사항을 모두 만족한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06



[그림 40] 제안된 소프트웨어의 use case diagram

개발된 소프트웨어는 파일 중 사용자에게 악성행위 코드를 추출하는 방법론을 설명하는 기능이 있어야 한다. 사용자가 파일을 분석하기 위해 파일을 직관적인 UI(user interface)로 업로드 할 수 있어야 하며, 업로드된 파일을 분석한 결과를 웹에서 확인할 수 있어야 한다. 이때 악성행위를 수행하는 코드가 파편화되어 있는 경우를 대비하여 탐지된 악성행위 코드를 목록화하여 사용자가 쉽게 결과를 확인할 수 있어야 한다. 분석된 결과를 사용자가 파일로 저장할 수 있어야 한다.

각 기능에 대한 구현 결과는 다음과 같다.

1. 파일 업로드
2. 분석 결과 시각화
3. 분석 결과 저장
4. 분석 방법 소개

웹 결과 그림과 함께 상세 내용 추가 예정

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

2.2.3 시스템 비기능(품질) 요구사항

계획서의 시스템 비기능(품질) 요구사항은 다음과 같으며, 본 소프트웨어는 계획서의 비기능 요구사항을 모두 만족한다.

1. 보안

악의적인 사용자가 대용량 파일을 반복하여 업로드하면 서비스가 지연되거나 서버가 다운되는 문제가 발생할 수 있다. 따라서 동일한 IP 에서 1 시간 내에 업로드 할 수 있는 파일의 수는 10 개, 파일의 크기는 300MB 로 제한한다.

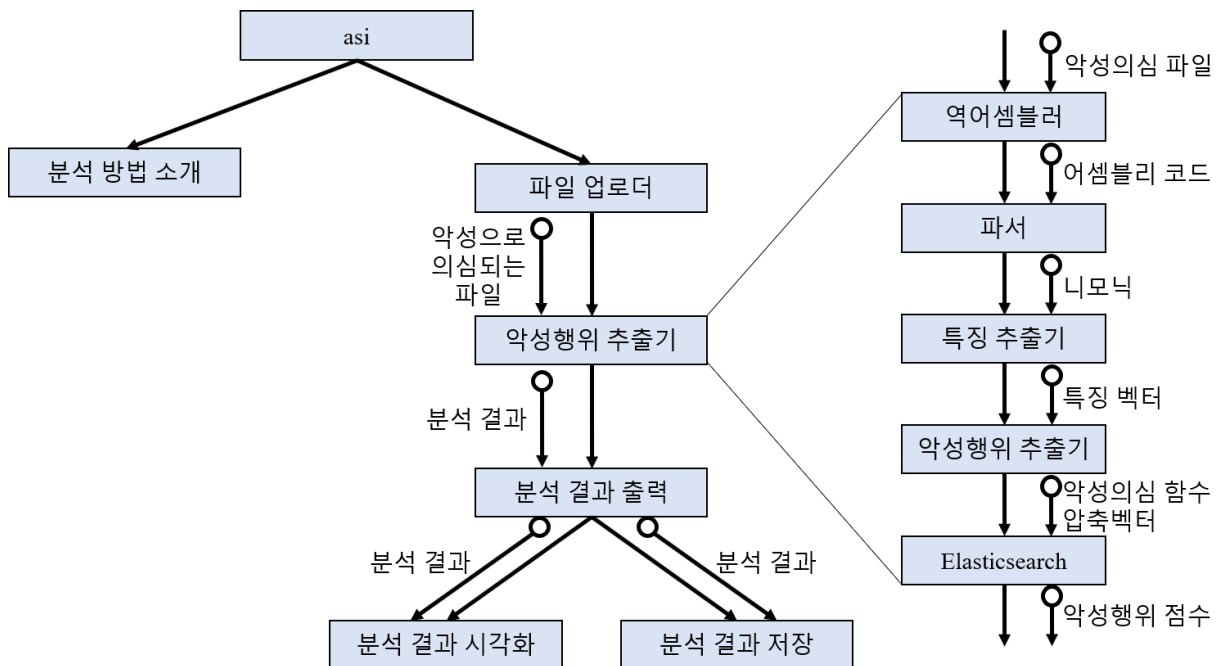
2. 성능

입력된 파일의 각 니모닉별 악성행위 여부를 빠르고 정확하게 판단할 수 있어야 한다. 제안된 소프트웨어는 1초에 2,000개 이상의 니모닉을 분석할 수 있어야 하며, 분석 결과는 전문가가 악성 행위라 판정한 코드의 70% 이상을 포함하고 있어야 한다.


웹 & 전문가보고서 비교 결과 확인 후 상세 내용 추가 예정

2.2.4 시스템 구조 및 설계도

제안된 소프트웨어의 시스템 구조도는 [그림 41]과 같다.




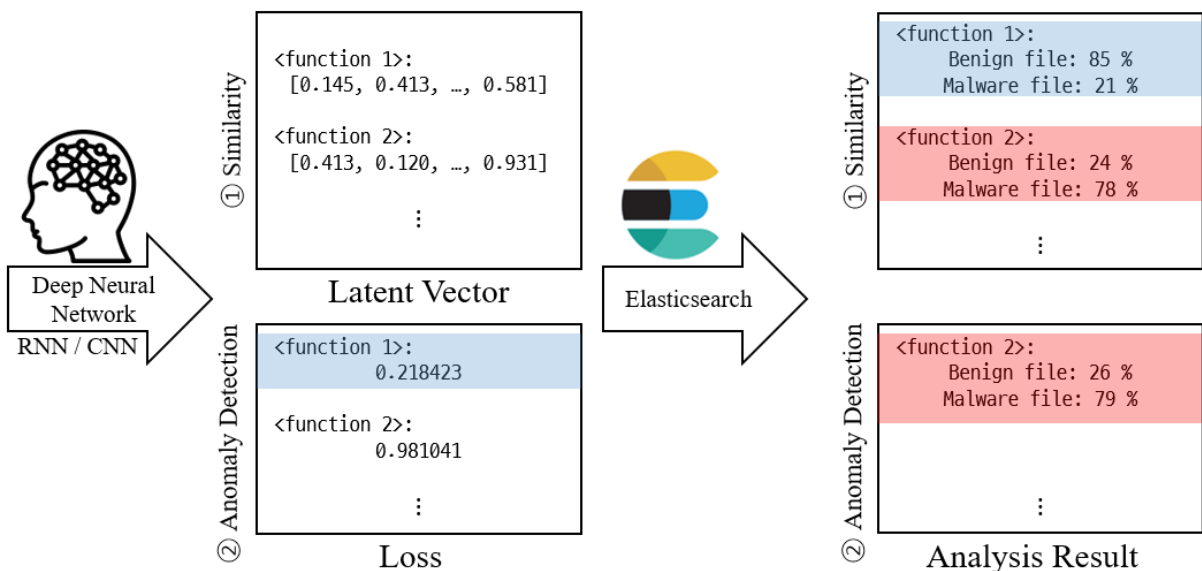
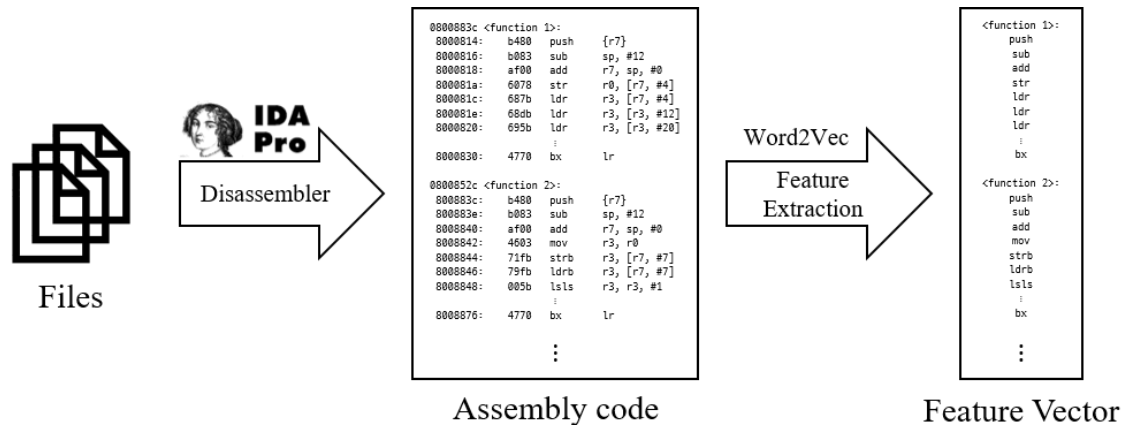
[그림 41] 제안된 소프트웨어의 시스템 구조도

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

각 모듈에 대한 설명은 다음과 같다. 분석 방법 소개 모듈은 우리가 제안한 소프트웨어의 방법론과 사용방법 기술한 문서를 보인다. 파일 업로더 모듈은 직관적인 드래그 앤 드롭(drag-and-drop) 방식과 파일 탐색기를 이용한 업로드 방식을 제공하여 사용자가 쉽게 파일을 서버에 업로드 할 수 있다.

악성행위 추출기는 역어셈블러, 파서, 특징 추출기, 악성행위 추출기, Elasticsearch 5 개의 모듈로 구성되어 있으며, 각 함수의 악성행위 수행 예상 결과를 점수화 한다. 역어셈블러는 IDA Pro 를 이용하여 악성으로 의심되는 파일을 역어셈블하여 어셈블리 코드를 추출한다. 본 프로젝트에서는 어셈블리 코드 중 니모닉만을 필요로 하므로 Python 기반으로 파서를 자체 개발해 니모닉을 추출한다. 특징 추출기에서는 Word2Vec 임베딩 알고리즘을 활용해 니모닉을 적합한 특징 공간으로 변환한다. 악성행위 추출기는 상술한 것과 같이 두 가지 방식으로 구현되었는데, 오토인코더 기반의 방법을 활용할 경우 악성의심 함수를, 압축벡터 기반의 방법을 활용할 경우 압축벡터를 반환한다. Elasticsearch 는 각 입력을 이용해 입력 함수와 유사한 함수들을 찾고, 각 함수들을 포함하는 파일의 정상/악성 여부에 따라 악성행위 수행 가능성을 점수화 하여 반환한다. 이상의 과정을 정리하면 [그림 42]와 같다.

 <div> 국민대학교 컴퓨터공학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06




[그림 42] 세부 분석 모듈

2.2.5 활용/개발된 기술

2.2.5.1. 인공 신경망

인공 신경망은 입력 벡터에 적합한 가중치 벡터를 내적하고, 비선형함수인 활성화함수(activation function)을 거치는 연산을 적용함으로써 뉴런(neuron)을 흉내 낸 것이다. 깊은 신경망은 이러한 신경망들을 결합함으로써 은닉층에서 적합한 특징을 추출하고, 출력층에서 특징벡터로부터 적합한 출력을 계산한다.

인공 신경망 중 가장 기본적인 신경망은 MLP(multi-layer perceptron)으로 퍼셉트론으로 이루어진 층 여러 개를 연결하여 출력을 계산한다. 그러나, MLP 는 가중치가 너무 많기 때문에 과적합(overfitting)이 발생할 가능성이 높다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

CNN(convolutional neural network)는 사진과 같이 지역적 특성이 있는 데이터에 적합한 신경망으로 특징벡터 중 일부분에 대한 내적을 수행하고, 가중치를 공유하는 기법을 활용해 이동에 둔감한 특성을 가지며 MLP의 과적합 문제를 완화한 신경망이다.


RNN(recurrent neural network)는 음성이나 심박수와 같이 시간적 특성이 있는 데이터에 적합한 신경망으로 이전 데이터에 대한 중간 결과를 저장하고, 이를 현재 데이터 연산에 활용함으로써 이전 결과를 기억할 수 있는 능력을 갖는다.

우리는 뉴모닉이 언어와 같이 시간적, 지역적 특성이 있음에 주목하여 CNN과 RNN을 기반으로 오토인코더를 구현하여 신경망이 뉴모닉의 언어모델을 학습하도록 구현했다. 이때 오토인코더는 입력과 동일한 출력을 갖도록 하는 신경망을 의미하며, 오토인코더가 입출력을 동일하게 하기 위해서는 인코딩 결과가 입력의 특징을 갖고 있어야 하므로, 인코딩 결과를 압축벡터로 사용할 수 있다. 또한, 정상파일의 언어모델을 학습한 RNN이 제대로 예측하지 못한다면, 이는 입력 함수가 정상파일의 언어모델에 적합하지 않음을 시사하므로 악성행위를 수행할 것이라 예측할 수 있다.

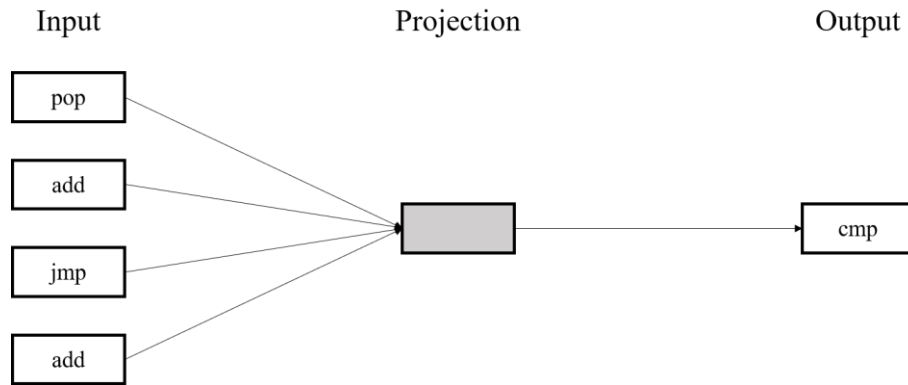
2.2.5.2. 단어 임베딩

자연어를 신경망으로 적합하게 학습시키기 위해서는 각 단어를 적합한 특징벡터 공간의 벡터로 변환하는 과정, 즉 단어 임베딩 과정을 적절히 수행해야 한다. 우리는 자연어와 유사한 특징을 갖는 어셈블리어를 분석하므로 입력 뉴모닉을 적합한 특징벡터로 변환하기 위해 Word2Vec, Doc2Vec, FastText 알고리즘을 적용하여 임베딩 기술을 확보했다.

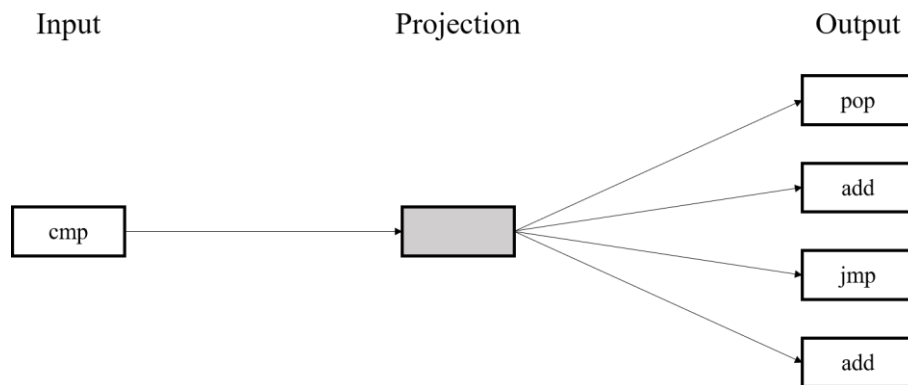
Word2Vec은 가장 많이 사용되는 임베딩 방법론 중 하나로, 기존 원 핫 인코딩(one-hot encoding) 방식과 다르게 단어의 개수보다 낮은 차원의 벡터로 단어를 임베딩할 수 있으며, 유사한 단어를 가까운 공간에 사상(mapping)시키는 방식이다. 예를 들어 사과, 배, 서울 세 단어를 임베딩한다면 원 핫 인코딩은 단어간의 관계를 고려하지 않고 일괄적으로 $(001)_2, (010)_2, (100)_2$ 로 임베딩하지만, Word2Vec은 사과와 배는 과일이라는 동일한 범주에 있으므로 가까운 공간에, 서울은 먼 공간에 사상시키는 방식이다. Word2Vec은 주변 단어를 이용해 중간 단어를 예측하는 방식인 CBOW(continuous bag of words) 방식과 중간 단어로 주변 단어를 예측하는 방식인 Skip-Gram 방식이 있다. 예를 들어 연산자가 pop, add, cmp, jmp, add 이고, 윈도우 사이즈가 2이며, cmp 명령어에 대해 학습하는 단계라면 CBOW는 [그림 43]과 같이 주변 연산자인 pop, add, jmp, add를 입력 받아 cmp를 예측하도록 학습하며, Skip-Gram은 [그림 44]와 같이 중심 연산자인 cmp를 입력 받아 주변 연산자인 pop, add, jmp, add를 예측하도록 학습한다. 일반적으로 Skip-Gram 방식이

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

CBOW 방식보다 효과적이라 알려져 있으며, 우리의 실험 결과 역시 Skip-gram 이 CBOW 에 비해 더 효과적으로 임베딩을 수행함을 확인했다.




[그림 43] CBOW 학습 방안



[그림 44] Skip-Gram 학습 방안

Word2Vec 학습시 고려해야 할 주요 하이퍼 파라미터는 출력 벡터의 차원, 윈도우 크기, 최소 단어의 수, 에포크(epoch), 학습률(learning rate)이다. 출력 벡터의 차원은 임베딩 결과로 나오는 벡터의 차원을 의미하고, 윈도우 크기는 학습에 사용하는 주변 단어의 개수를 의미한다. 위의 그림은 윈도우 크기가 2 인 경우이다. 최소 단어의 수는 등장 빈도가 적은 단어를 임베딩 시키지 않기 위해 필요하며, 이 숫자보다 단어가 적게 등장하는 경우 임베딩 시키지 않는다. 에포크와 학습률은 일반적인 딥러닝에서의 의미와 동일하다. 에포크는 전체 데이터의 학습을 반복하는 횟수를, 학습률은 오류 역전파(back propagation)시 가중치를 변화시키는 비중을 의미한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

Doc2Vec 은 Word2Vec 의 변형으로 학습의 입력으로 단어뿐만 아니라 문서의 색인을 추가로 받는다. Doc2Vec 은 각 문서의 단어가 가까운 특징공간에 사상되는 것을 목적으로 한다. Word2Vec 과 유사하게 주변 단어를 입력 받아 중심 단어를 예측하는 DBOW(distributed bag of words) 방식과 중심 단어를 입력 받아 주변 단어를 예측하는 Distributed-Memory 방식이 있다.

FastText 는 Word2Vec 의 변형으로 단어의 부분이 일치하는 단어는 유사한 단어임을 이용한다. 이를 위해 단어를 작은 단위로 나누는 방법론을 채택한다. 예를 들어 imul 이라는 연산자가 있고, 단어를 나누는 단위가 3 이라면, imu, mul 으로 나누어진 단어와 전체 단어 imul 을 임베딩한다. 이후 mul 이라는 연산자를 임베딩할 때는 imul 과 가까운 특징 공간으로 사상될 수 있다. FastText 의 학습 방법은 Word2Vec 과 동일하다.

2.2.6 현실적 제한 요소 및 그 해결 방안


실행 파일(executable file)로부터 원본 코드를 복구하는 역공학(reverse engineering)을 방지하거나, 파일의 크기를 줄이기 위하여 실행 파일을 압축하는 패킹(packing) 기법이 개발되고 있다. 특히, 악성코드의 경우 자동 분석기의 분석을 피하거나 전문가의 분석을 어렵게 만들기 위하여 패킹을 적용하는 경우가 많다. 패킹이 적용된 경우 IDA와 같은 역어셈블링 도구를 이용해 어셈블리어를 복구하기 어려우므로 우리의 기법을 적용할 수 없다.

패킹된 파일을 역어셈블링하는 것은 우리의 연구범위를 벗어나므로, 우리는 패킹되지 않은 파일에 대해서만 학습을 수행했으며, 서비스 역시 패킹되지 않은 파일에 대해서만 제공한다.

2.2.7 결과물 목록

[표 4] 결과물 목록 및 상세 사양

대분류	소분류	기능	형식	기술문서
웹 서비스	분석 방법 소개	본 프로젝트의 소개와 분석 방법론을 보인 다.		없음
	파일 업로더	사용자가 악성이라 의심되는 파일을 서버에 업로드한다.	DLL/함수	없음
	분석 결과 시각화	분석한 결과를 웹에서 시각화하여 사용자가 악성행위 수행부를 쉽게 확인할 수 있도록 한다.		없음
	분석 결과 저장	분석한 결과를 저장할 수 있도록 한다.		없음
악성행위 코드 추출기	역어셈블러	IDA Pro를 기반으로 파일의 어셈블리 코드를 추출한다.	함수	없음


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

	<i>파서</i>	Python을 기반으로 어셈블리 코드로부터 니모닉을 추출한다.	함수	없음
	<i>특징 추출기</i>	입력된 니모닉을 적절한 특징 벡터로 임베딩한다.	함수	없음
	<i>RNN 기반 악성행위 수 추출기</i>	입력된 특징벡터를 분석하여 악성행위를 수행할 것으로 의심되는 위치를 추출한다.	신경망	없음
	<i>CNN 기반 압축벡터 추출기</i>	입력된 특징벡터를 분석하여 함수를 가장 잘 나타낼 수 있는 압축벡터를 추출한다.	신경망	없음
	<i>Elasticsearch 기반 유사 함수 탐색기</i>	유사한 함수 검색 및 해당 함수를 포함하는 프로그램의 악성/정상 여부 판정		없음
데이터 수집	<i>크롤러</i>	정상파일을 수집하여 데이터셋을 구축한다.	함수	없음

2.3 기대효과 및 활용방안

제안된 소프트웨어를 활용하면 전문가가 빠르게 악성코드를 분석할 수 있으므로 더 적은 인력으로 악성코드를 분석할 수 있으며 신종 악성코드에 빠르게 대처할 수 있다. 특히 시큐리티대응센터와 같이 신속하게 악성코드를 분석해야 하는 환경에서 큰 도움이 될 것으로 기대되며, 정보보안에 대한 투자가 적은 국내의 상황에서 각 기업의 정보보안에 대한 수준을 높일 수 있을 것으로 기대된다.

또한, 우리의 프로그램은 컴퓨팅 능력이 낮은 환경에서도 동작 가능하며 악성코드에 대한 사전 지식이 적어도 사용가능한 직관적인 UI를 갖는다. 즉, 악성코드 분석 전문가뿐만 아니라 입문자 역시 제안된 소프트웨어를 쉽게 사용할 수 있다. 악성코드 분석 입문자는 분석 실습을 위해 우리의 프로그램을 사용함으로써 악성코드의 패턴과 분석 노하우를 학습할 수 있으므로 보안 전문가의 양성에 큰 도움이 될 것으로 기대한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06


3 자기평가

본 프로젝트의 최종 결과물은 딥러닝 기반 함수 단위의 악성행위 수행여부 점수화 프로그램이다.

매일 350000 개의 악성코드가 새로 생성되는 반면, 국내의 정보보호에 대한 투자는 매우 적으며, 전문가는 악성코드 하나를 분석하는데 최소 몇 시간에서 최대 몇 주가 소요되므로 전문가의 악성코드 분석을 돕는 소프트웨어의 개발이 시급한 실정이다. 그러나, 최근의 연구동향은 입력 파일의 악성행위 수행 여부를 판정하는 자동 분류기 연구에 치중되어 있고, 악성코드 분석 보조도구에 대한 연구는 전무하다.


우리는 입력 파일의 함수단위로 악성행위 수행여부를 판정하는 방법론을 개발하였다. 제안된 프로그램은 최초로 악성코드 분석 보조도구라는 점에서 의의를 갖는다. 또한, 이 프로그램은 전문가의 분석 보고서에 준하게 악성 수행여부를 판정할 수 있으므로 향후 전문가들이 악성코드를 분석하는데 큰 도움이 될 것으로 기대된다. 이는 더 적은 인력으로 악성코드 분석이 가능하며 전문가가 신종 악성코드에 신속히 대응할 수 있다는 것을 시사한다.

서비스 제공 측면에서 이용자의 악성코드 분석에 대한 지식이 전무해도 프로그램을 사용할 수 있도록 직관적인 UI 를 적용했으며, 서비스에 대한 상세한 소개를 포함시켰다. 또한, 이용자의 컴퓨팅 환경이 낮을 수 있음을 대비하여 분석 서버에서 제안된 프로그램이 동작하도록 구현했다. 즉, 누구나 우리의 프로그램을 사용할 수 있는 환경을 구축했다.


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
[1]	기사	Malware Statistics	https://www.av-test.org/en/statistics/malware/	2020	AV Test	
[2]	논문	An overview of anomaly detection techniques: Existing solutions and latest technological trends	Computer networks. Vol. 51. No. 12.	2007	Animesh Patcha and Jung-Min Park	
[3]	논문	Malware Detection with Deep Neural Network Using Process Behavior	IEEE 40th Annual Computer Software and Applications Conference	2016	Shun Tobiyama, Yukiko Yamaguchi, Hajime Shimada, Tomonori Ikuse, Takeshi Yagi	
[4]	논문	A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting	Future Generation Computer Systems 85	2018	Hamed HaddadPajouh, Ali Dehghantanha, Raouf Khayami, Kim-Kwang Raymond Choo	
[5]	논문	정적 분석 기반 기계학습 기법을 활용한 악성코드 식별 시스템 연구	한국정보보호학회 Vol. 29. No. 4	2019	김수정, 하지희, 오수현, 이태진	
[6]	논문	정적 분석과 앙상블 기반의 리눅스 악성코드 분류 연구	한국정보보호학회 Vol. 29. No. 6	2019	황준호, 이태진	
[7]	논문	악성코드 DNA 생성을 통한 유사 악성코드 분류기법	한국정보보호학회 Vol. 23. No. 4.	2013	한병진, 최영한, 배병철	
[8]	논문	Convolutional Neural Network 기반의 악성코드 이미지화를 통한 패밀리 분류	한국정보보호학회 Vol. 26. No. 1.	2016	석선희, 김호원	
[9]	논문	Deep Learning and Visualization for Identifying Malware Families	IEEE Transactions and Dependable and Secure Computing	2018	Guosong Sun, Quan Qian	
[10]	논문	악성코드 패밀리 분류를 위한 API 특징 기반 앙상블 모델 학습	한국정보보호학회 Vol. 29. No. 3.	2019	이현종, 여성울, 황두성	
[11]	기사	드라마 ‘유령’ 속 악성코드, 실제로는?	https://www.ahnlab.com/kr/site/securityinfo/secu news/secuNewsView.do?cmd=print&seq=19768&menu_dist=3	2012	AhnLab	
[12]	보고서	2018년 정보보호 실태조사	https://www.msit.go.kr/web/msipContents/contentsView.do?cateId=_stat us&artId=1513388	2018	과학기술정보통신부, 한국정보보호산업협회	
[13]	보고서	Malware Analysis Market by Component (Solution (Static Analysis and Dynamic Analysis) and Services), Organization Size (SMEs and Large Enterprises), Deployment (Cloud and On-premises), Vertical, and Region - Global Forecast to 2024	https://www.marketsandmarkets.com/Market-Reports/malware-analysis-market-108766513.html	2019	Markets and Markets	

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

[14]	사이트	ASEC 소개	https://www.ahnlab.com/kr/site/securityinfo/asec/asecIntro.do?	2020	AhnLab	
[15]	사이트	ESRC 소개	https://www.estsecurity.com/	2020	EST Security	
[16]	사이트	Microsoft Malware Prediction	https://www.kaggle.com/c/microsoft-malware-prediction	2018	Kaggle	
[17]	사이트	정보보호 R&D 데이터 챌린지 2019	http://datachallenge.kr/challenge19/rd-datachallenge/malware/introduction/	2019	한국인터넷진흥원	
[18]	논문	A Neural Probabilistic Language Model	Journal of Machine Learning Research 3	2003	Yoshua Bengio, Rejean Ducharme, Pascal Vincent, Christian Jauvin	
[19]	사이트	VirusTotal homepage	https://www.virustotal.com/gui/home	2020	VirusTotal	
[20]	사이트	Intezer homepage	https://intezer.com/	2020	Intezer	
[21]	사이트	V3 소개	https://www.ahnlab.com/kr/site/product/productView.do?prodSeq=15	2020	AhnLab	
[22]	사이트	What is Cuckoo	https://cuckoo.readthedocs.io/en/latest/introduction/what/	2020	Cuckoo Sandbox	
[23]	사이트	Joe Sandbox Cloud	https://www.joesecurity.org/joe-sandbox-cloud#overview	2020	Joe Security	
[24]	사이트	Elasticsearch 소개	https://www.elastic.co/kr/elasticsearch/	2020	Elastic	
[25]	사이트	IDA Pro 소개	https://www.hex-rays.com/products/ida/	2020	Hex-Rays	
[26]	사이트	gensim 라이브러리 소개	https://radimrehurek.com/gensim/	2020	gensim	
[27]	사이트	Keras 라이브러리 소개	https://keras.io/ko/	2020	Keras	

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

5 부록


5.1 사용자 매뉴얼

웹 결과 그림과 함께 상세 내용 추가 예정

5.2 테스트 케이스

[표 5] 테스트 케이스 목록

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
웹 서비스	파일 업로드	사용자가 악성이라 의심되는 파일을 서버에 업로드한다.	파일을 업로드하고 서버에서 동일한 파일이 업로드 되었는지 diff 명령어로 확인한다.	동일한 파일이 업로드 되어 diff 출력이 NULL이어야 한다.	성공
	분석 결과 시각화	분석한 결과를 웹에서 시각화하여 사용자가 악성행위 수행부를 쉽게 확인할 수 있도록 한다.	서버에서 동일한 파일을 분석한 결과가 정확히 시각화 되었는지 확인한다.	두 결과는 동일해야 한다.	성공
	분석 결과 저장	분석한 결과를 저장할 수 있도록 한다.	서버에서 동일한 파일을 분석한 결과 파일과 웹에서 다운로드한 파일이 동일한지 확인한다.	두 파일이 동일해야 한다.	성공
악성행위 코드 추출기	RNN 기반 악성함수 추출기	입력된 특징벡터를 분석하여 악성행위를 수행할 것으로 의심되는 위치를 추출한다.	각 방법을 적용하여 Elasticsearch로 유사도를 검색한 결과와 전문가의 분석 보고서를 비교한다.	전문가가 악성 행위를 수행한다고 분석한 함수의 대부분을 악성 행위를 수행할 것이라 예측해야 한다.	성공
	CNN 기반 압축벡터 추출기	입력된 특징벡터를 분석하여 함수를 가장 잘 나타낼 수			성공


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

		있는 압축벡터를 추출한다.			
	Elasticsearch 기반 유사 함수 탐색기	유사한 함수 검색 및 해당 함수를 포 함하는 프로그램의 악성/정상 여부 판 정			성공
데이터 수집	크롤러	정상파일을 수집하 여 데이터셋을 구 축한다.	크롤링을 지시한 위 치의 파일 목록과 크 롤러가 다운로드한 파일을 비교한다.	지시한 위치의 모든 파일을 크롤러가 다 운로드 한다.	성공

5.3 임베딩 알고리즘별 실험 결과

[표 6] Word2Vec(CBOW)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉


Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
mov	jmp lea jz jnz jle jbe call push jge jl	jmp jle jnz xor jbe jz jge jl jg ldarg.1	jnz jmp jle jz jg xor jge jbe jl cmpunordps	jmp jle jnz jz jg xor jge lea jbe jl	jmp jz jnz jle jge lea jl xor jg push
jmp	mov jnz jl jz jbe jge jg jle lea call	mov jnz ret jz lea jl ldarg.1 box jle jge	jnz mov jz jg jle lea jge setnz setz call	mov lea jnz jz call setz jge jle jl jg	mov lea jnz jz call jge jl push jle jg
add	jbe ja sub jnb jg jle jge jz jnz jb	ldloc.2 imul sub adc stloc.3 ldloc.3 ldloc.0 ja stloc.1 sar	ldc.i4.1 imul ldc.i4.3 ldc.i4.2 jg ja stelem.i4 ldelem.i4 sub bge.s	imul ja jb jnb jg jno adc sub jbe jl	jb ja jg jnb imul sub jl jbe adc shl

	국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
		프로젝트 명	asi(a security insight)	
		팀 명	assist(a security safety important special team)	
		Confidential Restricted	Version 1.0	2020-JUN-06

pop	leave retn cgt.un pushf stosd setns vzeroupper fsincos ldnull fclex	leave retn callvirt castclass brfalse.s isinst ldelem.ref blt.s stloc.2 brtrue.s	retn leave ldsfld callvirt ldelem.ref xor pushf stloc.s vzeroupper mov	leave retn vfmsub231ss divpd vfmsub213sd vpminsd vpackusdw movdq2q pextrb pmovzxwd	retn leave mov xor leave.s jmp vcmpgtsd cmpsd vdivss call
push	lea call mov jmp test throw ldtoken vfmsub231sd jl jge	lea call jz jnz mov jle jmp stloc.0 ldarg jge	call lea jz jnz js jmp box jle dup ldc.i4.0	lea call jz jnz jmp jle mov js setnz test	lea call jz jnz jmp mov jle test js jl

[표 7] FastText(Skip-Gram)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉


Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
mov	setz cmovnz cmovz setnz setns xor setnl sets setl test	lea push add vmovsd shlx shl cmp xor inc shl jno	lea xor push shlx add shl sar inc setnl cmovz	lea push xor add xchg inc setnz vmovsd cmp sar	lea push add xor inc cmp cmovz movzx movsd setz
jmp	jl jbe jb jnz jge jnb ja jle jg fsincos	jl jge jb jnb ja jbe jg jle retn jnz	retn jl jle jge jb jg jnb jbe jnz js	retn jle jg jl jge jns jnz lea js jb	retn jle jl jnz jge jg jns js jz jb
add	dec cmpneqsd cmovbe setnbe stmxcscr ucomisd btr cvtss2si maxpd cmp	sub prefetcht0 cmpltpd prefetcht1 mov movups adc cmpnlepd movddup prefetchnta	sub prefetcht0 prefetcht1 shl prefetch sar unpckhpd imul mov prefetchnta	sub shl mov prefetcht1 prefetcht0 lea cmp imul prefetch pmulldq	sub mov imul lea cmp shl movaps xorps addpd mulpd

	국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
		프로젝트 명	asi(a security insight)	
		팀 명	assist(a security safety important special team)	
		Confidential Restricted	Version 1.0	2020-JUN-06

pop	leave popf pushf jecxz sti stc vroundsd vzeroupper cld loopne	retn retf leave vzeroupper cmpltd popfw cmpjesd pushf xor les	retn popfw cmpsw xor roundsd cmpsd popf vzeroupper vroundsd vcmpgtd	retn leave push vroundsd popf cld setz roundsd vpbroadcastw popfw	retn push leave mov call lea cmpjesd vzeroupper jmp setz
push	jns js lea fnclex call stosd test cmovs setns jz	lea mov setns fnclex haddpd fsincos cmovns cmovnz cmovle cmovz	lea mov test fnclex setns cmovs setnz fldl xor setz	lea mov test setnz setz cmovnz stosd xor cmovns ficom	lea mov setnz setz test stosd cmovnz jz xor setns

[표 8] FastText(CBOW)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉


Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
mov	jmp jg jl cmovnz jnz jge jb setz jbe ldloc.3	jmp jle jbe jnz jge push jz jnb jl lea	jle jnz jmp jge jz jl jg jbe jnb jb	jle jnz jz jmp jge jg xor jbe jnb jl	jmp jnz jle jz jge jl jg jnb push jbe
jmp	mov jg jl ldloc.3 cmovnz jnz jge jb ldloc.2 jbe	mov jl jb jnz jge jbe fcmovnbe lea fcmovnb jg	mov jge lea jl jnz jg jle jb pushfw ja	mov lea jge jnz movss setp push jg jle jnb	mov lea movss jge jnz jl push movaps test jz
add	ldind.ref ldelem ldelema stmxcscr movzx ldelem.r8 ldelem.r4 endfinally stloc starg.s	movddup movshdup movsldup inc imul movntq movd movnti movsx movntdq	jno imul adc psignw movnti cvtps2dq fcmovb cvtps2dq paddusw pavgw	jno shlx jg imul adc movnti fiadd cmovb fimul movntdq	imul jg adc fimul fiadd addsd jno jl sub ja
pop	leave	retn	retn	retn	retn

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

	newobj retn ldnull ldvirtftn rethrow ldsflld ret leave.s iret	leave popfw popcnt brtrue popf fsave lds callvirt brtrue.s	leave popfw popcnt popf popa castclass leave.s isinst pushf	leave popfw leave.s popf movhlps popcnt br.s movlhps ldsflld	leave popfw leave.s popf popa popcnt xor br.s ldsfllda
push	lea call jl jmp mov ldelem.ref jle jge ldstr jnz	call lea ldtoken mov ldstr pushfw box calli jnz pusha	lea call pushfw calli jmp jnz jz jz pusha pmulhw mov	lea call pushfw pusha jnz jz jmp mov jle jge	lea call pushfw jmp jnz jz mov jl pusha test

[표 9] Doc2Vec(distributed-memory)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉


Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
mov	lea stosd xor push stosw movsd cmpxchg sar fldz stosb	lea push xor movss cmpsb xorps cmpsd sar fst jecxz	push movss pushf xchg vmovsd movq lea bswap paddsb cpuid	push lea cpuid aesenc movaps cmp xadd movss xchg imul	push lea vfmsub213sd vfmsub231ss vfmsub231sd cmp movzx vfmadd213ss vpclmulqdq vfmsub213ss
jmp	jno retn jnb jb jl ja jbe ror jge ud2	retn vpmuldq vfmsub213ps vhaddps retf ldloc.2 jl hlt vpmullw loope	retn jns js jle jnz jg ja jnb jb pxor	retn jnz js jle jns jz jl jb jnb jbe	retn jnz js jz jle jns jl jb jnb jge
add	sub switch cmpltss cbw ldc.i4.8 ldc.i4.4 and ldc.i4.1 vldmxcsr ldc.i4.6	vbroadcastsd vmovaps vmovddup prefetcht0 prefetchnta pavgb vbroadcastss vunpcklpd vmovq vmovhps	conv.i mul mul.ovf.un vmovntdq inc add.ovf ldind.u2 pshufb vmovdqu fadd	ldelem.i2 vaddss vmovntdq vminps add.ovf vcvtps2dq vmaxps vmovlps bne.un.s vpackuswb	add.ovf blt.un rem.un blt.un.s ldelem.i2 sub.ovf bge.un vpaddb ble.un div.un
pop	leave sfence endfinally	leave vpsrld cmpltss	leave jecxz retn	leave vhaddps jecxz	leave jmp callvirt

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

	vzeroupper lodsd pushf jecxz btrue bfalse ror	cvtpd2pi vpslldq vpmuldq vfmsub213ss vzeroupper vfmadd132sd vcmpgtps	castclass retf vzeroupper cmpltd loop jo iret	cvttps2pi retf loope andnpd vfmsub213ps iret vcmpgt_oqps	vpmuldq jecxz cvttps2pi iret loop retn hlt
push	lea mov call fldz setnle stosd ldtoken fucom fst fstp	lea mov pmulhw vcmpnle_uqps movsx vcmpgt_oqps psubsw movq psubw vfmadd213sd	mov lea paddsb movss test paddusb vaesenc subps movdqa vpxor	mov vfmsub213ss lea vfmsub213sd vsubss vfmadd213sd vfmsub213sd maxps packssdw vmaxss	mov vfmsub213sd vfmsub213sd lea vfmadd213sd vcmpgt_oqps vfmsub213ss cmpnleps maxps minps

[표 10] Doc2Vec(DBOW)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
mov	lea stosd xor push stosw movsd movsdb cmplxchg sar fldz stosb	lea push xor movss cmpsb xorps cmpsd sar fst jecxz	push movss pushf xchg vmovsd movq lea bswap paddsb cpuid	push lea cpuid aesenc movaps cmp xadd movss xchg imul	push lea vfmsub213sd vfmsub213ss vfmsub213sd cmp movzx vfmadd213ss vpclmulqdq vfmsub213ss
jmp	jno retn jnb jb jl ja jbe ror jge ud2	retn vpmuldq vfmsub213ps vhaddps retf ldloc.2 jl hlt vpmullw loope	retn jns js jle jnz jg ja jnb jb pxor	retn jnz js jle jns jz jl jb jnb jbe	retn jnz js jz jns jl jb jnb jge
add	sub switch cmpltd cbw ldc.i4.8 ldc.i4.4 and ldc.i4.1 vldmxcsr ldc.i4.6	vbroadcastsd vmovaps vmovddup prefetcht0 prefetchnta pavgb vbroadcastss vunpcklpd vmovq vmovhps	conv.i mul mul.ovf.un vmovntdq inc add.ovf ldind.u2 pshufb vmovdqu fadd	ldelem.i2 vaddss vmovntdq vminps add.ovf vcvtps2dq vmaxps vmovlps bne.un.s vpackuswb	add.ovf blt.un rem.un blt.un.s ldelem.i2 sub.ovf bge.un vpaddb ble.un div.un
pop	leave sfence endfinally vzeroupper lodsd	leave vpsrld cmpltd cvtpd2pi vpslldq	leave jecxz retn castclass retf	leave vhaddps jecxz cvttps2pi retf	leave jmp callvirt vpmuldq jecxz

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.0	2020-JUN-06

	pushf jecxz brtrue brfalse ror	vpmuldq vfmsub213ss vzeroupper vfnmadd132sd vcmpeqps	vzeroupper cmpltsd loop jo iret	loope andnpd vfmsub231ps iret vcmpgt_oqps	cvttps2pi iret loop retn hlt
push	lea mov call fldz setnle stosd ldtoken fucom fst fstp	lea mov pmulhw vcmpnle_uqps movsx vcmpl_oqps psubsw movq psubw vfmadd231sd	mov lea paddsb movss test paddusb vaesenc subps movdqa vpxor	mov vfmsub231ss lea vfmsub231sd vsubss vfmadd231sd vfmsub213sd maxps packssdw vmaxss	mov vfmsub231sd vfmsub213sd lea vfmadd231sd vcmpgt_oqps vfmsub231ss cmpnleps maxps minps