Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.4, Aug. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.4.735

소스코드 취약성 분류를 위한 기계학습 기법의 적용*

이 원 경,[1†] 이 민 주,[1] 서 동 수[2‡]
[1,2]성신여자대학교(대학원생, 교수)

Application of Machine Learning Techniques for the Classification of Source Code Vulnerability*

Won-Kyung Lee,[1†] Min-Ju Lee,[1] DongSu Seo[2‡]
[1,2]Sungshin University(Graduate student, Professor)

요 약

시큐어코딩은 악의적인 공격 혹은 예상치 못한 오류에 대한 강인함을 제공해줄 수 있는 안전한 코딩 기법으로 정적분석도구의 지원을 통해 취약한 패턴을 찾아내거나 오염 데이터의 유입 가능성을 발견한다. 시큐어코딩은 정적기법을 적극적으로 활용하는 만큼 룰셋에 의존적이라는 단점을 가지며, 정적분석 도구의 복잡성이 높아지는 만큼 정확한 진단이 어렵다는 문제점을 안고 있다. 본 논문은 시큐어코딩을 지원하는 목적으로 기계학습 기법 중 DNN과 CNN, RNN 신경망을 이용하여 개발보안가이드 상의 주요 보안약점에 해당하는 패턴을 학습시키고 분류하는 모델을 개발하며 학습 결과를 분석한다. 이를 통해 기계학습 기법이 정적분석과 더불어 보안약점 탐지에 도움을 줄 수 있을 것으로 기대한다.

ABSTRACT

Secure coding is a technique that detects malicious attack or unexpected errors to make software systems resilient against such circumstances. In many cases secure coding relies on static analysis tools to find vulnerable patterns and contaminated data in advance. However, secure coding has the disadvantage of being dependent on rule-sets, and accurate diagnosis is difficult as the complexity of static analysis tools increases. In order to support secure coding, we apply machine learning techniques, such as DNN, CNN and RNN to investigate into finding major weakness patterns shown in secure development coding guides and present machine learning models and experimental results. We believe that machine learning techniques can support detecting security weakness along with static analysis techniques.

Keywords: Secure Coding, Static Analysis, Machine Learning

I. 서 론

최근 활발히 연구되고 있는 기계학습은 다양한 분야로 그 적용범위를 넓혀가고 있다. 기계학습의 핵심은 방대한 양의 데이터를 바탕으로 유의미한 수준의 학습을 수행함으로써 사람이 행하는 판단을 확률적으로 시뮬레이션 한다는 점이다. 특히 자율주행자동차[1], IoT, 스마트 팩토리[2] 등 이미지로부터 학습된 결과를 적용하는 분야에 효과적인 것으로 알려졌으며 대규모 데이터에 의존한 응용분야에 성공적으로 적용되고 있다. 방대한 양의 데이터로부터 학습이 가능하다는 것은 이미지 데이터 뿐 아니라 자연어와 같은 규칙성을 사람의 대화를 모사하는 챗봇과 같은

Received(06. 12. 2020), Modified(07. 01. 2020), Accepted(07. 03. 2020)
* 이 논문은 2017년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음
* 이 논문은 2018년도 이원경의 석사논문을 개선 및 확장한 것임
† 주저자, good9103@gmail.com
‡ 교신저자, dseo@sungshin.ac.kr(Corresponding author)

대화생성 분야에도 활용되고 있다[17].

시큐어코딩(secure coding)은 악의적인 공격이나 오염된 데이터에 의해 유발될 수 있는 시스템의 보안성 침해를 선제적으로 방어하는 목적의 코딩 기술로서 공공서비스 뿐 아니라 안전 중요시스템이나 보안중요 시스템의 핵심 코딩기술로 인식되고 있다[3][4]. 시큐어코딩의 특징 중 하나는 안전하다고 알려진 코딩 스타일 뿐 아니라 다양한 형태의 취약한 패턴을 개발자가 학습함으로써 코딩활동에 활용한다는 점이다. 실제 개발 활동에서는 방대한 양의 소스코드와 광범위한 코딩 규칙으로 인해 많은 경우 정적분석 도구의 사용을 필요로 한다.

본 논문에서는 시큐어코딩 활동을 하는 과정에서 생성된 텍스트 데이터를 기계학습을 통해 학습시킴으로써 개발자의 시큐어코딩 활동이 기계학습을 통해 학습이 가능한지를 알아보고자 한다. 특히 빈번하게 발견되는 보안취약점 중 크로스사이트 스크립트 공격, 널 포인터 역참조 에러 등의 소스코드 내의 보안 취약점을 중심으로 기계학습을 진행하여 학습의 특성을 관찰하고 결과를 분석한다.

본 논문은 취약성 패턴의 학습을 위해 비선형적인 예측이 가능한 세 가지 신경망으로서 심층 신경망과 합성곱 신경망, 순환신경망을 활용한다. 이를 통해 어떤 신경망 모델이 적절한지를 찾아내기 위해 다양한 지표들과 파라메터, 활성함수를 활용하여 실험하며 정확도를 중심으로 한 실험 결과를 제시한다.

논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 소개하며, 3 장에서는 소스코드의 보안성과 관련한 정적기법과 기계학습 기법을 설명한다. 4 장에서는 실험환경 설명과 실험결과를 분석하며, 5 장에서는 결론과 더불어 향후 발전 방향을 논한다.

II. 관련연구

정적분석을 시큐어코딩에 활용하는 시도는 오랜 기간에 걸쳐 진행되어왔으며 이를 지원하는 다양한 분석도구[5]와 개발보안 가이드[6]와 같은 문서들이 개발되었다. 정적분석은 도구의 특성에 따라 패턴탐지, 경로분석, 심볼 수행(symbolic execution)등 다양한 기법을 활용하고 있다.

최근 들어 소스코드를 분석하기 위한 수단으로 기계학습 기법을 활용하는 연구가 보고되고 있다. 기계학습을 소스코드 분석에 적용시킨 연구는 크게 품질에 관한 연구와 취약성에 관한 연구로 나누어 볼 수 있다. Barstad와 Goodwin[7]는 오픈소스를 대상으로 K-최근접 이웃(K-nearest neighbour)알고리즘, 베이지안 분류(Bayesian classifier)을 적용하여 학습하고 이 결과를 수동적 리뷰와 비교하였다.

Russell[8]은 C와 C++ 소스코드를 대상으로 강화학습을 적용하여 취약성의 분류를 수행한 바 있다. 이 연구에서는 비교적 탐지가 수월한 버퍼 오버플로우 패턴을 탐지하였으며, 랜덤 포레스트 기법과 딥러닝 기법을 적용하였다.

Kolosnjaji[9]는 심층신경망을 사용하여 소스코드 내의 호출 순서를 학습시킴으로써 악성 데이터가 전파될 수 있는 경로의 존재를 탐지할 수 있는 연구를 수행한 바 있다. Zhen Li[10]는 이미지 데이터에서 주요 개체의 특징을 집중시키는 Pre-training 기법인 Attention과 BLSTM을 이용하여 C/C++ 언어로 작성된 소스코드의 취약점을 탐지하며, 취약점 종류를 분류를 수행한다.

본 논문은 Russell, Kolosnjaji 등의 연구와 맥을 같이하여 소스코드에 내재되어 있는 기계학습을 기반으로 한 취약점 탐지의 목적을 두고 있으나, Java 코드를 대상으로 국내 실정에 맞도록 행정안전부 개발보안가이드에 나타난 주요 보안약점을 분류 대상으로 하는 차별화된 접근을 한다. 이는 국내 시큐어코딩 및 개발보안 활동에 기계학습을 적용하는 실험적인 시도로 의미가 있으며 학습 과정에서 다양한 학습모델을 이용함으로써 취약성 패턴의 학습 가능여부를 살펴보고자 한다.

III. 정적분석과 기계학습

3.1 정적분석 기법

정적분석은 프로그램 실행을 기반으로 오류를 탐지하는 테스팅 기법과는 달리 비실행 기반의 소스코드 특성분석 분석기법이다[10]. 그러한 이유로 정적분석은 소스코드에 유입된 보안 약점 및 취약점을 자동 분석하는 기법으로 활용되고 있다.

정적분석 도구는 방대한 양의 소스코드에 대해 내부의 구조를 검사하기 위해 전처리 작업을 필요로 하며, 전처리 작업으로 구문 분석과 구문트리 생성을 한다. 구문트리는 코드의 구문특성을 구조적으로 표현한 것으로 정적분석에서는 이를 대상으로 제어흐름 분석과 자료흐름분석 등을 통해 다양한 정보를 추출한다.

정적분석은 사전에 정의된 취약한 패턴 혹은 규칙을 이용하거나 키워드 분석 등 비교적 단순한 탐지도 하지만 실행기반 테스팅에 버금가는 탐지능력을 제공하여 보안취약점 탐지에 효과적인 방법으로 인식되고 있다. 시큐어코딩을 위해 사용되는 대표적인 정적분석의 종류는 Table 1.과 같다.

국내의 경우 행정안전부는 개발보안가이드를 통해 공공기관의 정보시스템 구축에서는 정적분석 및 자동화된 도구의 이용을 권하고 있다.

Table 1. Static Analysis Techniques

Techniques	Description
Pattern Matching	It registers vulnerability pattern in vulnerability databases. Vulnerability can be found through pattern matching processes
Lexical Analysis	It searches illegal keywords or patterns by using of lexical analysis, particularly useful for coding style
Type Checking	It checks type dismatch in source code between type declaration and variable usages.
Data Flow Analysis	It uses control flow analysis over abstract syntax trees and inter-procedural call graphs to identify illegal data flows

3.2 기계학습 기법

기계학습의 출발점은 방대한 데이터로부터 얻어진 학습결과를 통해 분류 문제를 해결할 수 있는가 하는 점이다. 단일 신경망과 같이 은닉층이 1개로 구성되는 경우 선형적인 추론에 의한 분류가 가능하지만 뛰어난 식별해야 될 경우, 혹은 분류 유형의 많아질 경우는 단일경망으로는 정확한 결과를 얻기 힘들다. 이러한 문제에 대해 심층신경망 (Deep Neural Network, DNN)은 입력층과 출력층을 제외한 퍼셉트론 노드가 2개 이상의 은닉층을 정의하여 학습시킴으로 답을 찾고자 한다(Fig. 1.).

심층 신경망의 각 은닉층은 이전 계층의 출력 값을 다음 계층의 입력 값으로 전달하는 구조를 가지고 있다. 이런 구조로 인해 데이터의 고유한 특징을 추상화할 수 있으며, 학습 과정에서 자동으로 특징추출이 이루어지며, 나아가서는 데이터 간의 비선형적인 관계를 예측함으로써 단일신경망으론 할 수 없던 식별 문제를 해결한다.

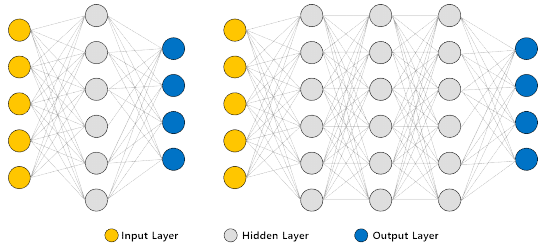


Fig. 1. Comparison of Single Neural Networks and Deep Neural Networks

일반적으로 심층 신경망은 크기가 작은 이미지 학습에 대해서는 문제없이 동작하지만, 이미지의 크기가 커질 경우, 혹은 학습에 필요한 피처의 집합이 방대해질 경우 학습에 많은 시간이 걸린다는 단점이 있다. 또한 이미지 내의 개체의 위치와 크기, 방향, 혹은 기울기가 변경될 경우 변화를 식별을 못하는 문제가 발생한다. 합성곱 신경망(Convolutional Neural Network, CNN)은 이러한 문제를 해결하기 위해 제안된 모델로 심층 신경망과 함께 이미지 인식 분야에 효과적인 것으로 알려졌다.

합성곱 신경망은 시신경 피질의 뉴런이 국소영역에 위치한 시각 자극에만 반응하여 하위레벨의 정보를 조합한 후 상위 레벨로 전달함으로써 계층적인 패턴으로 식별한다는 아이디어에 착안한 모델이다. 합성곱 신경망은 입력된 데이터를 축약하여 하위 계층에서 상위 계층으로 전달하기 위해 특징을 추출하는 합성곱(convolution) 활동과 자료를 축약하는 풀링(pooling) 활동을 반복 수행한다(Fig. 2.).
일반적으로 합성곱 신경망은 합성곱 계층과 풀링 계층을 복수로 정의한 형태를 가짐으로서 심층 신경망이 해결하지 못했던 이미지의 외곡, 위치변화, 축소확대 등으로부터 발생한 식별 문제를 해결한다.

심층신경망과 합성곱 신경망과는 다른 방식의 접근을 하는 순환신경망(Recurrent Neural network, RNN)[16]은 자연어와 같은 시퀀스 정보를 학습하는데 효과적인 학습모델로 알려져 있다.

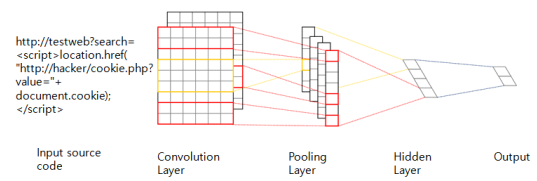


Fig. 2. Structure of Convolutional Neural Network,

순환신경망은 직전의 은닉상태를 기억하여 다음 스텝의 상태를 결정하는 순환과정을 활용하며 단어 간 혹은 문장 간의 시퀀스를 학습한다. 짧은 자연어 분석에는 효과적이지만 긴 문장의 경우 학습이 진행됨에 따라 기울기 소실 문제로 인해 앞 문장의 정보를 잃어버릴 위험이 있다. 이를 보완하기 위한 모델로 순환과정 상의 은닉상태 기억을 제어하는 게이트 정보를 추가한 LSTM(Long Short-term Memory)이 사용되고 있다.

IV. 취약패턴 학습 실험

4.1 실험환경의 구성

이미지 식별에 효과를 보이는 심층 신경망과 합성곱 신경망이 순환신경망과 유사하게 텍스트 형태의 데이터에 대해서도 효과적이라는 점은 여러 사례를 통해 보고되고 있다[11]. 본 논문의 실험에서는 소스코드와 같은 잘 정의된 구조를 갖는 텍스트 데이터에 대해서도 학습이 가능하다면 Fig. 3.과 같이 기계학습을 활용해서도 정적분석의 일정 부분기능을 수행할 수 있음을 보이고자 한다.

신경망을 훈련시키기 위해서는 학습데이터 혹은 피처 셋(feature set)을 정의하는 일이 선행되어야 한다. 텍스트 정보의 연속체로서 소스코드를 학습 데이터로 사용하기 위해서는 두 가지 경우가 가능하다. 첫째, 라인단위의 소스코드를 텍스트 형태로 학습에 사용하는 경우와, 둘째, 토큰 단위의 전처리를 거쳐 벡터데이터로 학습하는 경우이며, 본 실험에서는 학습 정확성 문제로 토큰단위의 학습을 수행한다.

본 실험에서 사용하는 데이터는 정적분석기를 통해 보고된 리포트를 통해 추출하였으며 보안취약점이

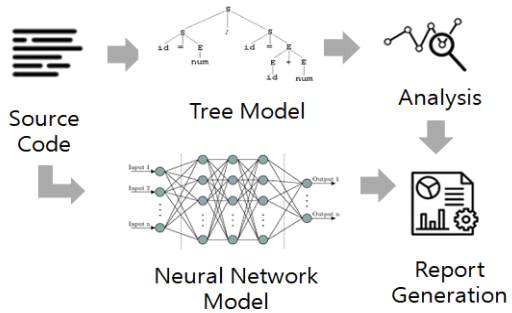


Fig. 3. Use of Neural Networks for Static Analysis

발생 코드 부분(code fragment)과 취약성 카테고리가 포함된 정보를 활용한다. 주로 보안 취약점이 발견된 소스코드 라인의 부분과 그에 해당하는 보안 약점 유형을 결과는 정답 라벨로 활용할 수 있어 지도 학습 모델로 훈련시킬 수 있다 (Fig. 4.).

실험을 위해 사용된 데이터는 한국인터넷진흥원에 의해 수집되고 제공된 총 68,863개의 취약점 목록으로 해당 취약점이 발생한 소스코드를 포함한다. 본 실험은 내재되어 있는 보안 취약점의 유형을 탐지하고 분류하는 것이 목표인 만큼 주어진 이들 데이터 중에서 정탐으로 판별된 소스코드 데이터만을 사용하였다. 이 중에서 신경망의 훈련에 쓰일 수 있을 만큼의 충분한 개수를 가진 크로스 사이트 스크립트, 널 포인터 역참조, 부적절한 예외처리, 시스템 데이터 정보 노출 등 총 4 영역의 유형을 분류하여 실험하였다. 실험에 사용된 학습환경은 Google Brain Team의 텐서플로우이며, Python 2.7 버전을 사용하여 모델을 구현하였다.

학습 데이터는 텐서플로우 API의 기본 단위인 텐서로 변환되어야 한다. 텐서는 다차원 데이터 배열의 형태를 가지며 데이터 플로우 노드의 입력으로 사용된다. 수치 연산을 나타내는 노드 간의 사이를 연결하는 것이 텐서의 역할로 텐서는 정수, 실수와 같은 수치 데이터로 표현되어야 한다. 이를 위해 본 실험에서는 텐서에 해당하는 소스코드 문자열 스트림을

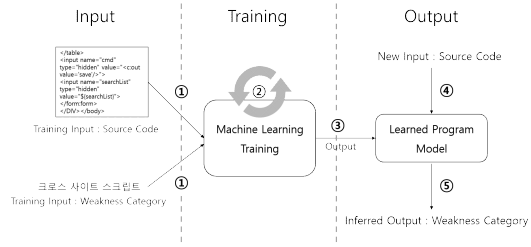


Fig. 4. Procedures for Machine Learning

Table 2. Data Size for Vulnerability Categories

Vulnerability Category	Training	Validation	Testing	Total
Cross site scripting	14,014	1,557	100	15,671
Null pointer error	9,604	1,067	100	10,771
illegal exception handling	6,590	732	100	7,422
system information exposure	3,855	428	100	4,383

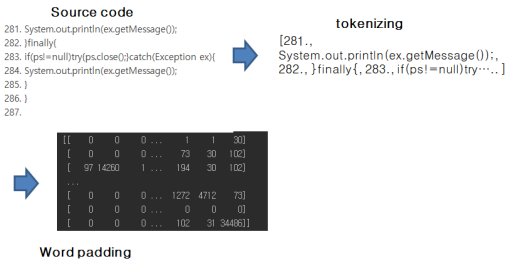


Fig. 5. Transformation of Source Code Tensor

숫자로 변환해주기 위해 Fig. 5.와 같은 단계로 나누어 소스코드 스트림을 텐서로 변환한다.

심층 신경망을 이용한 보안 취약점 식별 실험에서 신경망은 총 3개의 은닉층으로 정의되었으며 각 은닉층은 512개의 노드로 설계되었다. 입력층은 가장 많은 토큰으로 이어진 소스코드의 개수를 기준으로 노드 개수가 결정된다. 실험에서는 가장 긴 토큰 개수인 801개를 기준으로 입력층의 노드 개수 정의되었다. 출력층 노드는 식별해야 하는 보안 취약점 유형을 나타내므로 4개로 구성되며, 각 노드는 크로스 사이트 스크립트, 널 포인터 역참조, 부적절한 예외처리, 시스템 정보노출을 표현한다. Fig. 6.은 심층 신경망 실험의 구조를 보여주기 위해 도식화한 모형으로 입력-은닉-출력의 계층과 노드들을 보여준다.

심층신경망 구조에서는 어떤 활성화 함수를 사용하느냐에 따라 오류율과 정답률이 달라진다. 본 실험에서는 대표적인 비선형 활성화 함수인 Sigmoid, Tanh, 선형 활성함수인 Relu6를 적용하여 결과를 비교한다. 실험에서 설계된 합성곱 신경망은 48,289 x801개의 입력데이터에 대해 3x128의 구조를 갖는 합성곱 층과 맥스 풀링층, 그리고 완전연결계층(fully conneted layer)과 분류기로 구성된다.(Fig. 7.)

정규화는 데이터를 정답 곡선 안에 포함시키기 위

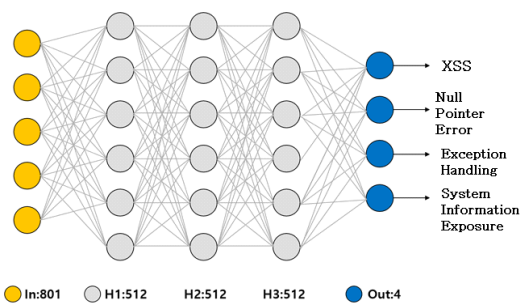


Fig. 6. Structure of Empirical DNN Model

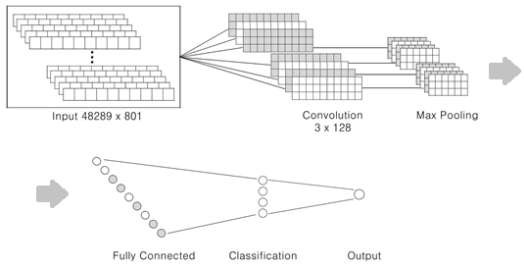


Fig. 7. Structure of Empirical CNN Model

해 과하게 굴곡졌던 신경망 모델을 펴주는 방법으로 데이터의 특징 개수를 줄여 모델을 단순하게 만들어 주며 과적합(overfitting)을 피하도록 하는 이점이 있다. 정규화를 하는 방법에는 L1 정규화와 L2 정규화가 있으며, 두 개의 방식은 가중치에 제약을 거는 수식에서 차이가 있다. L1 정규화는 절대 값의 합을 적용하여 상수와 같은 영향이 미미한 작은 가중치들은 0으로 수렴이 되게 만들고 중요한 몇 개의 가중치들만 영향을 미치게 만드는 함수라면, L2 정규화는 모든 파라미터에 제곱만큼의 크기로 가중치에 제약을 두어 몇 개의 입력 데이터에 강하게 가중치를 적용하기 보단 모든 입력 데이터에 영향력을 약하게 적용되게 만드는 함수이다. (수식 1)과 (수식 2)는 각각 L1 정규화와 L2 정규화의 수식을 나타낸다.

$$S = \sum_{i=0}^{n} \left| y_i - h(x_i) \right| \tag{1}$$

$$S = \sum_{i=0}^{n} (y_i - h(x_i))^2 \tag{2}$$

4.2 실험 결과

기계학습의 효율성을 검증하기 위한 지표로 본 논문에서는 학습 정확도, 검증정확도, 그리고 테스트 정확도 지표를 사용한다. 테스트 정확도는 훈련과 검증을 거쳐 만들어진 기계학습 모델의 정확도를 판별하기 위한 목적으로 훈련에 사용되지 않은 새로운 보안 취약점 데이터를 유형별로 100개 입력했을 때 정답일 확률로 정의한다.

손실함수(loss function)는 학습모델로부터 생성된 예측과 정답사이의 차이를 판단하는 함수로 손실이 최소가 되는 부분을 찾아가도록 최적화되며 교차

엔트로피(corss-entropy) 함수를 이용해 정의된다. 실험에서는 6개의 은닉층을 정의하였으며 학습비율은 0.0001, 드롭아웃 계수는 0.3, Epoc은 10을 적용하였으며 Adam optimizer를 사용하였다.

심층신경망의 실험결과 Table 3.과 같이 활성화 함수 Tanh를 사용했을 경우가 훈련데이터에 대한 Accuracy로 0.79를 보였으며, 검증 데이터에 대해서는 0.74를, 테스트 데이터에 대해서는 0.83의 정확도를 보였다. 결과적으로 Tanh는 4개의 활성화 함수에 대한 모델 중 가장 높은 테스트 정확도를 보인다.

Table 3. Accuracy Table for DNN

Activation function	Accuracy / Loss	Training Average	Verification average	Test Acc.
Sigmoid	Accuracy	0.66	0.65	0.62
	Loss	0.83	0.83	
Tanh	Accuracy	0.79	0.74	0.83
	Loss	0.53	0.66	
Relu6	Accuracy	0.77	0.73	0.79
	Loss	0.85	0.68	

Fig. 8. Accuracy(a) and Loss(b) for DNN Sigmoid Function

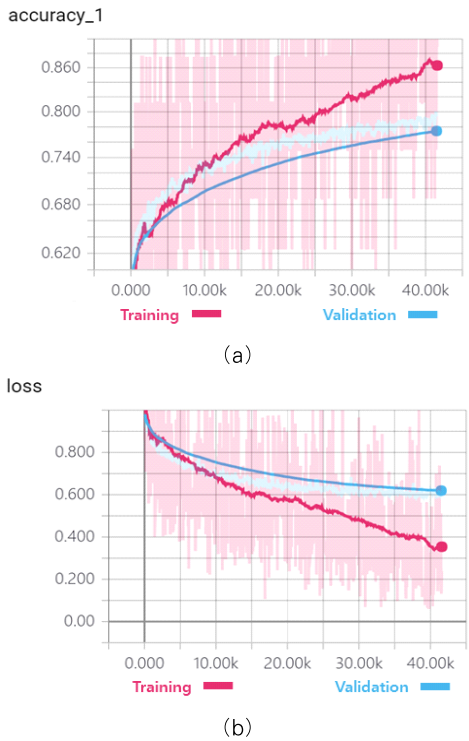


Fig. 9. Accuracy(a) and Loss(b) for DNN Tanh Function

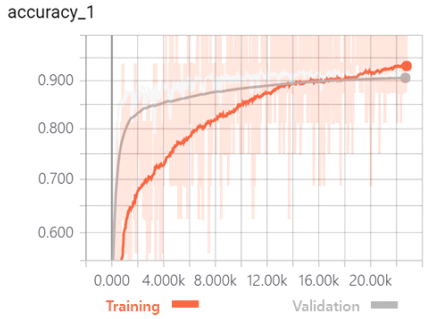
동일한 방식으로 두 번째 실험으로 얻어진 합성곱 신경망의 정확도는 Table 4.와 Fig. 10.을 통해 보여진다. L1 정규화와 L2 정규화, 그리고 기본(default) 정규화 기법에 따라 적용된 합성곱 신경망의 학습 정확도를 보여주며, 학습과정은 활성화 함수로 기본함수 경우만 그림으로 제시한다.

RNN과 LSTM의 경우 의 경우 드롭아웃 0.3, 20 Epoc을 적용하여 실험한 결과 카테고리 분류에는 유의미한 실험결과를 얻지는 못하였으나 이진분류

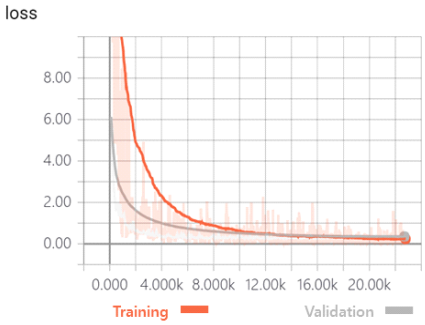
Table 4. Accuracy Table for CNN

Activation function	Accuracy / Loss	Training Average	Verification average	Test Acc.
Default	Accuracy	0.94	0.92	0.87
	Loss	0.2303	0.2372	
L1 Function	Accuracy	0.82	0.89	0.78
	Loss	0.0548	0.0543	
L2 Function	Accuracy	0.93	0.92	0.80
	Loss	1.41E-05	1.229E-05	

의 경우 XSS에 대해서는 95.1%, 널포인터 역참조 오류에 대해서는 97.3%의 정확성을 보이는 것을 관찰할 수 있다.



(a)



(b)

Fig. 10. Accuracy(a) and Loss(b) for CNN Default Function

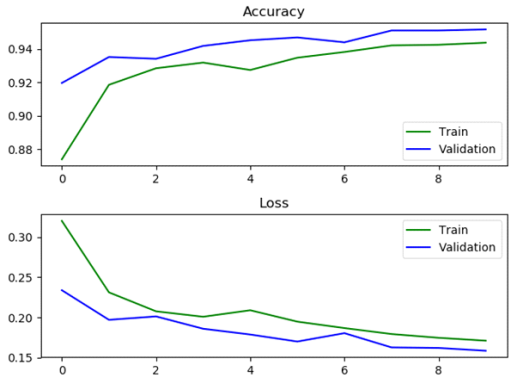


Fig. 11. Accuracy(a) and Loss(b) of LSTM for XSS

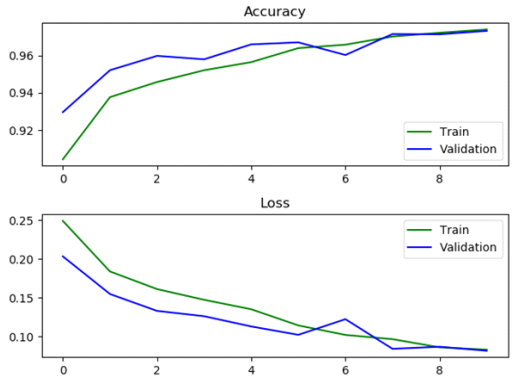


Fig. 12. Accuracy and Loss of LSTM for Nullpointer Dereferencing

V. 결 론

본 연구를 통해 얻어진 주요 내용을 정리하면 다음과 같다.

첫째, 활성함수의 수준에 따라 학습결과는 차이가 있지만 취약성 분류에 있어 전반적으로 합성곱 신경망 모델이 심층 신경망 모델에 비해 우수한 결과를 보이고 있다.

둘째, 실험데이터가 충분한 크로스사이트 스크립트의 경우 학습과정에서 큰 문제는 없었지만 상대적으로 적은 양의 데이터로부터 출발한 부적절한 예외처리, 시스템 데이터 정보노출 카테고리의 경우 과적합 문제가 발생할 수 있다고 판단된다. 실제 테이블 3의 테스트 정확도가 트레이닝 정확도를 넘어서는 경우가 그러한 이유인 것으로 생각된다.

셋째, 충분한 양의 학습데이터가 제공된다면 4개 영역 뿐 아니라 다른 영역의 카테고리까지 확장하여 식별 가능할 것으로 판단된다.

넷째, 가공되지 않은 데이터로 학습한 경우는 유의미한 수준의 정확도가 도출되지 않았다. 이를 개선하기 위한 본 연구에서는 토큰 단위의 구분을 통해 벡터화시키는 전처리를 수행하였으며, 결과적으로 의미있는 학습이 진행될 수 있음을 관찰하였다.

결론적으로 기계학습 기법 중에서 이미지 식별에 적합하다고 알려진 합성곱 신경망이 소스코드 취약점을 탐지하는 분야에도 적용이 가능하다는 점과 더불어 LSTM과 같은 순환모델을 활용한다면 다양한 측면의 접근이 가능하다고 볼 수 있다. 그럼에도 불구하고 알고리즘 기반인 기존의 정적분석과 비교해볼 때 아직은 대체할 수 있는 수준의 탐지 능력을 보여

주지는 못한 것이 현재의 기계학습 모델의 한계라 생각한다. 다만 앞으로 충분한 양의 학습데이터가 확보된다면 정교한 소스코드 취약성 탐지에 도움을 줄 수 있을 것으로 기대된다.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi view 3D object detection network for autonomous driving," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6526-6534, July. 2017.

[2] J. Wang, Y Ma, L. Zhang, R.X. Gao, and D. Wu, "Deep learning for smart manufacturing: methods and applications," *Journal of Manufacturing Systems*, vol. 48, part C, pp. 144-156, Jan. 2018.

[3] V.B. Livshits and M.S Lam, "Finding security vulnerabilities in java applications with static analysis," *Proceedings of the 14th conference on USENIX Security Symposium*, vol. 14, pp. 18-18, Aug. 2005.

[4] Y.W. Huang, F. Yu, C. Hang, C.H. Tsai, D.T. Lee, and S.Y. Kuo, "Securing web application code by static analysis and runtime protection," *Proceedings of the 13th international conference on World Wide Web,* pp. 40-52, May. 2004.

[5] A.M. Delaiter, B.C. Stivalet, P.E. Black, V. Okun, T.S. Cohen, and A. Ribeiro, "*Sate v report: ten years of static analysis tool expositions*," No. Special Publication, (NIST SP)-500-326, 2018

[6] Ministry of the Interior and Safety and Korea Internet & Security Agency, "Development security guide for sw developers and operators of e-government," 11-1311000-000330-10, Jan. 2017

[7] V. Barstad, M Goodwin, and T Gjoseter, "Predicting source code quality with static analysis and machine learning," *Norsk IKT-konferanse for forskning og utdanning*, Jan. 2015.

[8] R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, and M. McConley, "Automated vulnerability detection in source code using deep representation learning," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, pp. 757-762, Dec. 2018

[9] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," *Australasian Joint Conference on Artificial Intelligence*, Springer, Cham, pp. 137-149, Nov. 2016.

[10] L. Zhen, Z. Deqing, X. Shouhuai, O. Xinyu, J. Hai, W. Sujuan, D. Zhijun, and Z. Yuyi, "Vuldeepecker : a deep learning-based system for vulnerability detection," *Proceedings 2018 Network and Distributed System Security Symposium*, 2018, Jan. 2018.

[11] B. Chess, and J. West, *Secure programming with static analysis*, Pearson Education, Jun. 2007

[12] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNL)*, 2014, pp. 1746-1751, Aug. 2014

[13] S. Christey, and R.A. Martin, "*Vulnerability type distributions in cve*," Mitre report, May. 2007

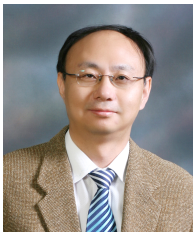
[14] J. Williams, and D. Wichers, "*The ten most critical web application security risks*," rc1, OWASP Foundation, 2017

[15] W.K. Lee, "A study on detection and classification of security vulnerabilities based on machine learning," MSc Thesis, Sungshin University, Aug. 2018
[16] R. Nallapati, B. Zhou, C.N. santos , C. Gulcehre, and B. Xiang. "Abstractive text summarization using sequence-to-sequence rnns and beyond," *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 280-290, 2016
[17] Y. Tom, H. Devamanyu, P. Soujanya, and C. Erik, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55-75, Aug. 2018

〈저자소개〉

이 원 경 (Won-Kyung Lee) 학생회원
2015년 8월: 성신여자대학교 IT학부 졸업
2018년 8월: 성신여자대학교 대학원 컴퓨터학과 석사
〈관심분야〉 기계학습, 소프트웨어공학, 악성코드 분석

이 민 주 (Min-Ju Lee) 학생회원
2018년 8월: 성신여자대학교 IT학부 졸업
2018년 9월~현재: 성신여자대학교 대학원 컴퓨터학과 석사과정
〈관심분야〉 기계학습, 자연어처리, 소프트웨어공학



서 동 수 (Dongsu Seo) 정회원
1986년: 중앙대학교 컴퓨터공학과 졸업(학사)
1990년 :영국 맨체스터대학교 대학원 Computation학과 졸업 (석사)
1994년: 영국 맨체스터대학교 대학원 Computation학과 졸업 (박사)
1998~현재 성신여자대학교 컴퓨터공학과 교수
〈관심분야〉 정보보호, 소프트웨어공학, 정형기법