The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 4, pp.111-117, Aug. 31, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.4.111
JIIBC 2022-4-16

딥러닝 기법을 사용하는 소프트웨어 결함 예측 모델

Prediction Model of Software Fault using Deep Learning Methods

홍의석*

Euyseok Hong*

요 약 수십년간 매우 많은 소프트웨어 결함 예측 모델에 관한 연구들이 수행되었으며, 그들 중 기계학습 기법을 사용한 모델들이 가장 좋은 성능을 보였다. 딥러닝 기법은 기계학습 분야에서 가장 각광받는 기술이 되었지만 결함 예측 모델의 분류기로 사용된 연구는 거의 없었다. 몇몇 연구들은 모델의 입력 소스나 구문 데이터로부터 시맨틱 정보를 얻어내는데 딥러닝을 사용하였다. 본 논문은 3개 이상의 은닉층을 갖는 MLP를 이용하여 모델 구조와 하이퍼 파라미터를 변경하여 여러 모델들을 제작하였다. 모델 평가 실험 결과 MLP 기반 딥러닝 모델들은 기존 결함 예측 모델들과 Accuracy는 비슷한 성능을 보였으나 AUC는 유의미하게 더 우수한 성능을 보였다. 또한 또다른 딥러닝 모델인 CNN 모델보다도 더 나은 성능을 보였다.

Abstract Many studies have been conducted on software fault prediction models for decades, and the models using machine learning techniques showed the best performance. Deep learning techniques have become the most popular in the field of machine learning, but few studies have used them as classifiers for fault prediction models. Some studies have used deep learning to obtain semantic information from the model input source code or syntactic data. In this paper, we produced several models by changing the model structure and hyperparameters using MLP with three or more hidden layers. As a result of the model evaluation experiment, the MLP-based deep learning models showed similar performance to the existing models in terms of Accuracy, but significantly better in AUC. It also outperformed another deep learning model, the CNN model.

Key Words : Fault prediction, Deep learning, Machine learning

Ⅰ. 서 론

소프트웨어 결함 예측 모델은 결함이 발생할 문제 부분들을 미리 예측함으로써 적절한 자원할당, 효율적인 프로세스 개선 등을 통해 전체 개발 프로젝트 성공을 가능케한다. 수십년간 소프트웨어 결함 예측 분야는 매우 많은 연구가 진행되었으며, 기계학습 알고리즘들을 사용하면서 예측 성능에 상당한 진전을 보았다.

딥러닝 기법은 기계학습 분야에서 가장 각광받는 기술이 되었지만 소프트웨어 결함 예측 연구에 사용된 경우

*정회원, 성신여자대학교 컴퓨터공학과
접수일자 2022년 6월 15일, 수정완료 2022년 7월 15일
게재확정일자 2022년 8월 5일

Received: 15 June, 2022 / Revised: 15 July, 2022 / Accepted: 5 August, 2022
*Corresponding Author: hes@sungshin.ac.kr
Dept. of Computer Engineering,
Sungshin Women's University, Korea

는 그리 많지 않다. 특히 기존 예측 모델들이 분류기 제작을 위해 여러 기계학습 기법들을 사용한 것에 비해 딥러닝이 분류기 제작에 사용된 경우는 거의 없다. 예측 모델에 딥러닝이 사용된 경우의 대부분은 그림 1의 ①의 경우로 소프트웨어 설계나 소스 코드로부터 새로운 시맨틱 피쳐 집합을 얻어내는 데 사용되었다.

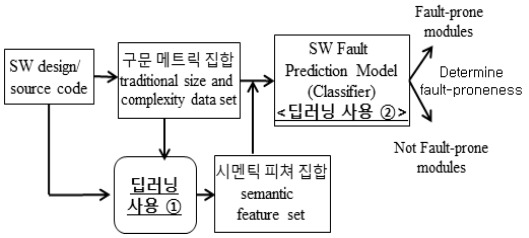

그림 1. 소프트웨어 결함 예측에 딥러닝 기법 사용
Fig. 1. Using deep learning methods for SW fault prediction

분류기 제작에 딥러닝을 사용한 연구가 거의 없는 이유는 소프트웨어 결함 데이터 집합의 크기가 딥러닝을 적용하기에는 너무 작고, 딥러닝 알고리즘들의 특정 적용 분야가 아니기 때문이다. 본 논문은 예측 모델에 적합한 형태의 딥러닝 기법을 사용하는 분류기들을 제작하여 예측 성능을 기존 모델들과 비교해보고 딥러닝이 결함 예측 분류기로 의미가 있는 기법인지 알아본다.

기존의 신경망 사용 예측 모델의 구조는 은닉층이 하나인 MLP 신경망이었다. MLP는 은닉층이 하나 이상인 퍼셉트론 신경망을 의미하지만 본 논문에서는 기존 신경망 모델을 MLPs(MLP with single hidden layer), 은닉층이 2개 이상인 딥러닝을 사용한 MLP 심층 신경망을 DNN(Deep Neural Networks)이라 지칭한다. 본 논문의 주된 목적 중 하나는 DNN을 사용한 예측 모델의 성능이 MLPs 등을 사용한 기존 모델의 성능보다 나아지는가를 알아보는 것이다.

2장에서는 기존 예측 모델 연구들을 살펴보고, 3장에서는 모델 구조 및 하이퍼 파라미터 설정에 대해 설명한다. 4장에서는 실험 데이터와 평가 척도 및 실험 결과를 언급하고, 5장에서 결론을 기술한다.

II. 결함 예측 모델

결함 예측 모델은 훈련 데이터집합의 출력값 즉 결함 여부의 유무에 따라 감독형, 비감독형, 세미감독형으로 분류된다. 지금까지 제안된 대부분의 예측 모델들은 감독형 모델들이었으며 로지스틱 회귀분석, MLPs, 베이지안 분류기, 판단 트리, SVM 등의 기법들을 사용하였다[1]. 라벨 데이터가 없는 경우 비감독형 모델 제작에 사용한 기법들은 K-means, EM, DBSCAN 등의 클러스터링 기법들이며[2], 일부만 라벨값이 존재할 경우 세미감독형 모델 제작에 사용된 기법들은 YATSI와 같은 Self-training 방법들이다[1].

기존의 예측 모델들이 사용한 데이터 집합은 주로 코드 사이즈나 복잡도에 기반한 구문 메트릭 집합이었다. 하지만 최근 몇몇 연구들은 딥러닝 기법을 사용하여 소스 코드로부터 시멘틱 피쳐 집합들을 얻어내어 예측 모델의 입력에 포함시켜 모델의 예측 성능을 높였다[3]. 이 연구들은 소스코드로부터 코드 토큰, AST(Abstract Syntax Tree) 정보를 꺼내어 딥러닝 기법에 적용하였으며 사용된 기법들은 DBN, LSTM, CNN, Transformer 모델들이 사용되었다[4].

[5]는 DNN과 CNN을 이용하여 결함 예측 모델의 분류기를 제작하였으며 여러 학습 파라미터들을 조정하여 예측 모델의 성능을 극대화할 수 있는 가를 연구하였다. 파라미터들은 에포크수, 배치사이즈, 드롭아웃율, 은닉층 수 등이며 데이터 집합은 NASA의 PROMISE 초기 데이터[4]를 사용하였고, 평가 척도로 Accuracy, TNR, TPR을 사용하였다. 하지만 성능 척도로 중요한 AUC를 사용하지 않았으며, 성능 실험 시 다른 파라미터 값들을 고정하고 하나의 파라미터 값만을 변경한 경우의 성능 비교를 했다는 문제가 있다. 가장 큰 문제점은 PROMISE의 5개 프로젝트에서 매우 작은 크기의 4개 프로젝트만 실험하여서 딥러닝 효용성 검증에 필요한 충분히 큰 데이터집합을 사용하지 않았다는 것이다. 본 논문은 이 프로젝트들 크기에 5~20배에 달하는 JM1을 사용한다.

표 1. NASA PROMISE 데이터 집합의 프로젝트 속성
Table 1. Project properties in the NASA PROMISE dataset

프로젝트	언어	모듈수	FP수	FP율
CM1	C	498	49	9.83
KC1	C++	2109	326	15.45
KC2	C++	522	105	20.50
PC1	C	1109	77	6.54
JM1	C	10885	2106	19.35

CNN은 메쉬 형태 구조를 갖는 데이터에 적합한 딥러닝 기법이다. 즉 2차원 구조에서 일정 위치의 데이터와

주변 데이터의 연관 관계를 유지하며 학습하는 구조로 일차원 벡터 형태의 데이터 구조인 예측 모델에 적합한 기법이 아니다. 하지만 [5]의 시험 결과는 CNN이 MLP에 기반한 DNN보다 훨씬 예측 성능이 좋았다.

Ⅲ. 모델 셀계

1. 모델 구조

은닉층이 여러개인 DNN의 구조를 결정하는 것은 기본적으로 은닉층의 개수와 노드의 수이다. 이들의 최적값을 결정하는 방법은 없으며 대부분 경험적인 방법으로 최적의 값을 찾아낸다. 일반적으로 단순한 DNN의 경우 2~5개의 은닉층으로 대부분의 문제를 해결할 수 있다. 본 연구에서 사용하는 결함 예측 모델의 데이터 크기가 이미지 등의 데이터에 비해 상대적으로 작으므로 은닉층의 수는 3, 5개로 하였다.

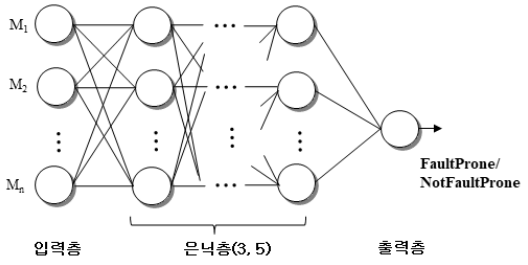

그림 2. 노드수가 같은 경우 모델 구조
Fig. 2. Model structure when the number of nodes is same

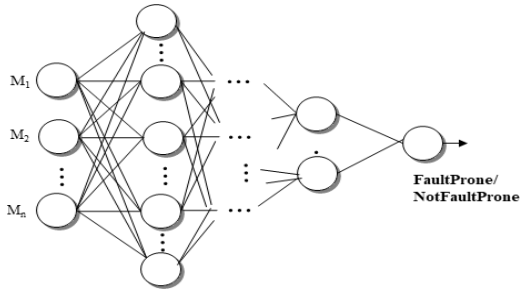

그림 3. 노드수가 다른 경우 모델 구조
Fig. 3. Model structure when the number of nodes is different

은닉층의 노드 수는 데이터 집합에 따라 입력층에 가까울수록 더 크게 정하는 경우도 있지만, 많은 경우 모든 은닉층의 노드수를 같게 한다. 본 연구는 두 형태를 모두 구현하였다. 그림 2와 그림 3은 입력 모듈 메트릭 벡터가 (M_1,M_2,...,M_n)인 n개의 입력을 받아 FP/NFP를 출력하는 이진 분류 모델의 구조를 표현한 것이다.

모든 은닉층 노드수가 같은 경우 노드수는 입력값 차원축소 여부에 따라 16, 32, 64와 32, 48, 64로 하였다. 입력층에서 출력층으로 갈수록 노드수가 감소하는 경우의 노드수는 은닉층 수와 차원축소 여부에 따라 32-16-8, 64-32-16, 64-64-48-32-16 등으로 하였다. 본 실험 데이터 사이즈가 다른 딥러닝 케이스보다 상대적으로 작기 때문에 너무 많은 노드 구조는 과적합의 위험이 커진다. 따라서 입력층 노드 수를 고려하여 여러 노드수의 경우를 실험하였다.

2. 하이퍼 파라미터

하이퍼 파리미터는 가중치나 바이어스와 같이 학습되는 모델 내부에서 결정되는 것이 아니라 모델 외부에서 사용자가 결정해주는 값들을 의미한다. 하이퍼 파라미터들의 최적값을 정하는 방법은 없고, 경험 또는 휴리스틱한 방법을 통해 최적값을 찾아내는 튜닝 작업을 거쳐 결정된다.

모델 예측치와 실제값 사이의 오차를 측정하기 위한 손실함수는 이진분류에서 일반적으로 사용하는 binary crossentropy 함수를 사용한다.

학습 시 가중치 변화도 양 극단에 있는 방법인 확률적 경사하강법이나 풀 배치학습이 아닌 일반적으로 많이 사용하는 미니배치 방법을 사용한다. 이 때 중요한 하이퍼 파라미터는 하나의 미니 배치에 넘겨주는 데이터 개수인 배치 사이즈이다. 딥러닝을 사용하는 경우에 큰 배치 사이즈를 사용하는 경우들이 있지만 큰 배치 사이즈는 지역최소화에 빠질 위험을 크게한다. 또한 작은 배치 사이즈를 사용하는 것이 광범위한 실험에서 주어진 계산 비용에 대해 좋은 훈련 안전성과 일반화 성능을 보인다. [7]은 여러 데이터 실험을 통해 32이하에서 가장 좋은 결과를 얻었으므로 본 연구에서는 배치 사이즈를 각각 5, 10, 32로 설정하였다.

학습에서 전체 데이터 집합의 학습과 테스트를 반복하는 횟수인 에포크 수의 결정도 중요하다. 과적합 문제로 훈련 정확도가 증가하더라도 검증 정확도는 감소하는 경우도 생긴다. [5]는 에포크 수를 10~20,000 사이값으로 실험하였고 10,000이 넘어가는 값에서 좋은 성능 결과를 얻었지만 본 연구의 실험 결과 에포크수가 200부터 과적합이 시작되고 10,000이 넘어가면 좋지 않은 결과

가 나타났다. 따라서 과적합 문제를 해결하고 불필요한 학습 로드를 줄이기 위해 Keras에서 제공하는 EarlyStopping 함수를 사용하였다. 이는 테스트 데이터 집합의 오류가 줄지 않으면 학습을 멈추게하는 콜백함수이다.

과적합을 막는 또다른 대표적인 기법인 드롭아웃도 사용한다. 드롭아웃은 몇 개의 노드들을 학습과정에서 랜덤하게 제외, 즉 출력을 0으로 하는 것이다. 이는 동일한 데이터에 대해 서로 다른 노드들이 학습에 참여하게함으로써 특정한 방향으로 데이터가 학습되는 것을 방지하는 기능을 한다. 드롭아웃 비율은 전체 노드 중에 출력을 0으로 하는 노드를 의미하며 보통 0.2~0.5를 사용한다. 본 논문의 실험에서는 0.5로 하였다.

신경망의 오류 역전파 단계에서 오류를 줄이기 위해 사용하는 최적화 함수는 일반적으로 많이 사용하는 adam을 사용하였다. 활성화 함수 역시 가장 일반적인 형태로 은닉층에서는 ReLU 함수를, 출력층에서는 sigmoid 함수를 사용하였다.

Ⅳ. 실험 및 결과

1. 데이터 집합 및 평가 척도

NASA MDP 데이터 집합은 13개의 프로젝트로 구성된 원본과 이를 정제하여 PROMISE 레포지토리에 올린 두가지 버전이 존재한다. PROMISE도 몇 개의 버전이 존재하지만 최근 PROMISE 사이트에는 표 1과 같이 5개의 주요 프로젝트만이 PROMISE 초기 버전 형태로 올라와있다[6]. 비교 연구인 [5]가 이들 중 CM1, KC1, KC2, PC1을 사용하였으므로 본 연구도 이 버전의 JM1을 사용하였다. JM1은 C언어로 구현된 실시간 시스템이며 프로젝트 사이즈가 315 KLOC인 대형 프로젝트로 나머지 4개 데이터 집합보다 매우 큰 빅데이터 집합이다. JM1의 또 다른 선정 이유는 다른 프로젝트에 비해 결측값이 적고, 프로젝트를 구성하는 파일들 사이의 불일치성이 매우 적기 때문이다[8].

실험에 사용할 JM1을 분석한 결과 5개 모듈 데이터에 명백한 문제가 있음을 발견하였다. 문제 모듈은 total_Op, total_Opnd 등의 값이 '?'로 나타난 모듈들이다. 따라서 JM1은 10,885개의 모듈로 구성돼있지만 본 논문의 실험은 10,880개의 모듈로 실험하였다.

JM1의 입력 메트릭은 McCabe의 복잡도 관련 메트릭들, Halstead의 볼륨 관련 메트릭들과 그외의 코드 사이즈 관련 메트릭들로 구성된다. 차원축소 작업은 예측 모델 연구에 가장 많이 사용되는 CFS(Correlation based Feature Selection)를 사용하였다. 표 2는 차원축소 결과를 나타낸 것이며 차원축소 경우를 FS, 모든 속성을 사용한 경우를 NFS로 나타내었다.

표 2. CFS의 차원축소 결과
Table 2. Dimensionality reduction results with CFS

차원축소 여부	입력 메트릭
NFS	loc, v(g), ev(g), iv(g), n, v, l, d, i, e, b, t, lOCode, lOComment, lOBlank, lOCodeAndComment, uniq_Op, uniq_Opnd, total_Op, total_Opnd, branchCount
FS	loc, v(g), ev(g), iv(g), i, lOComment, lOBlank, lOCodeAndComment

모델의 성능평가를 위한 척도로는 ACC(Accuracy), AUC를 사용하고 부가적으로 두 오류율인 FPR, FNR도 고려하였다. 결함 관련 데이터는 결함 모듈이 비결함 모듈보다 매우 적은 불균형 데이터이므로 전체 데이터 중에서 예측 성공 비율인 ACC만으로 모델 성능을 평가하긴 어렵다. 이를 보완하기 위해 가장 많이 사용하는 척도는 ROC curve 면적을 측정한 AUC이다. AUC는 1에 가까울수록 높은 성능을 나타내며 0.7 이상인 모델은 사용가능한 정확도 높은 모델로 분류된다[9]. FPR은 실제 비결함 모듈 중에 예측이 틀린 비율이고, FNR은 실제 결함 모듈 중에 예측이 틀린 비율로 FNR이 상대적으로 중요한 오류율이다.

2. 기존 모델 성능 평가 실험

DNN 모델과 성능 비교를 위해 딥러닝을 사용하지 않는 기존 모델들을 JM1을 사용하여 구현하였다. 기존 연구들에서 가장 많이 사용한 모델들을 사용하였으며 은닉층 하나인 MLPs와 결정트리(J48) 모델을 구현하였다. WEKA를 사용하여 구현하였으며 모델의 파라미터들은 기본값들을 사용하였다[10]. 표 3은 실험 결과를 나타낸 것으로, 5개 문제 모듈을 포함한 경우와 제외한 경우를 모두 실험하였으며 제외한 경우가 미세하게나마 나은 결과를 보였다. MLPs가 J48보다 나은 성능을 보였으며, J48은 FNR이 상대적으로 낮았으나 ACC, AUC 모두 MLPs보다 좋지 않았다. FNR이 심각하게 안좋은 결과를 보인 것은 JM1의 데이터 불균형 문제가 심각하다는 것을 의미한다.

표 3. 딥러닝을 사용하지 않은 모델 실험결과
Table 3. Experimental results of models without deep learning

모델	모듈수	ACC	AUC	FPR	FNR
MLPs	10,885	0.8096	0.690	0.010	0.943
	10,880	0.8112	0.696	0.016	0.910
J48	10,885	0.7950	0.653	0.070	0.768
	10,880	0.7954	0.653	0.071	0.762

3. 딥러닝 모델 성능 평가 실험

표 4와 표 5는 차원축소 유무에 따른 DNN 실험 결과이다. Loss는 손실함수 값을 나타낸다. 차원축소 유무에 따른 두 경우 ACC의 평균값은 0.8116과 0.8122로 차이가 거의 없다. AUC는 0.7187에서 0.7223으로 안한 경우가 0.0036 정도 좋은 결과를 보였지만 미미한 차이이므로 차원축소 여부는 성능에 큰 영향을 주지 않았다.

은닉층수는 성능에 영향을 미치지 않는 결과를 보인다. 배치사이즈의 증가는 차원축소 경우에는 결과에 영향을 미치지 않지만 모든 속성을 사용한 경우는 AUC가 증가하는 경향을 보였다. DNN의 ACC는 기존 모델인 MLPs와 유의미한 차이가 없었다. 하지만 AUC는 모든 실험 경우에 MLPs보다 높았다. DNN의 AUC 평균값은 MLPs 보다 0.026 증가하였다. 표 6은 은닉층의 노드수를 다르게 한 경우의 주요 결과를 나타낸 것으로 노드수를 같게 한 경우의 DNN 결과와 차이가 없다.

표 4. 딥러닝 모델 실험결과 - 차원축소
Table 4. Experimental results of DNN - FS

은닉층수	배치사이즈	노드수	Loss	ACC	AUC	FPR	FNR
3	5	16	0.4585	0.8099	0.7175	0.0087	0.9458
		32	0.4447	0.8101	0.7186	0.0034	0.9667
		64	0.8531	0.8097	0.7220	0.0413	0.8110
	10	16	0.4503	0.8097	0.7204	0.0025	0.9729
		32	0.4504	0.8122	0.7205	0.0038	0.9543
		64	0.4487	0.8135	0.7202	0.0074	0.9330
	32	16	0.4442	0.8102	0.7209	0.0055	0.9577
		32	0.4430	0.8134	0.7234	0.0052	0.9420
		64	0.4399	0.8121	0.7272	0.0061	0.9453
5	5	16	0.4589	0.8092	0.7176	0.0079	0.9525
		32	0.4726	0.8119	0.7013	0.0025	0.8996
		64	0.4876	0.8091	0.7031	0.0112	0.9774
	10	16	0.4522	0.8092	0.7194	0.0036	0.9705
		32	0.4467	0.8108	0.7210	0.0061	0.9520
		64	0.4456	0.8126	0.7235	0.0102	0.9259
	32	16	0.4356	0.8211	0.7123	0.0061	0.9623
		32	0.4434	0.8114	0.7224	0.0061	0.9491
		64	0.4419	0.8135	0.7247	0.0046	0.9439

표 5. 딥러닝 모델 실험결과 - 모든속성사용
Table 5. Experimental results of DNN - NFS

은닉층수	배치사이즈	노드수	Loss	ACC	AUC	FPR	FNR
3	5	32	0.4847	0.8132	0.7221	0.0190	0.8868
		48	0.5578	0.8109	0.7215	0.0303	0.8516
		64	0.5816	0.8124	0.7218	0.0260	0.8615
	10	32	0.4457	0.8124	0.7211	0.0042	0.9529
		48	0.4493	0.8125	0.7208	0.0066	0.9419
		64	0.4534	0.8130	0.7136	0.0068	0.9386
	32	32	0.4421	0.8128	0.7221	0.0063	0.9415
		48	0.4400	0.8124	0.7261	0.0072	0.9396
		64	0.4388	0.8130	0.7256	0.0100	0.9253
5	5	32	0.4549	0.8118	0.7210	0.0045	0.9543
		48	0.4572	0.8122	0.7240	0.0145	0.9101
		64	0.5655	0.8096	0.7192	0.0294	0.8621
	10	32	0.4530	0.8104	0.7230	0.0036	0.9652
		48	0.4448	0.8121	0.7244	0.0091	0.9339
		64	0.4548	0.8124	0.7174	0.0071	0.9405
	32	32	0.4421	0.8116	0.7251	0.0046	0.9548
		48	0.4411	0.8134	0.7259	0.0066	0.9377
		64	0.4410	0.8136	0.7260	0.0066	0.9367

표 6. 각 은닉층의 노드수를 다르게 한 경우 실험결과
Table 6. Experimental result when the number of nodes in each hidden layer is different

은닉층수/FS여부	배치사이즈	노드수 변화	Loss	ACC	AUC	FPR	FNR
3/FS	32	32-16-8	0.4437	0.8089	0.7212	0.0028	0.9757
3/NFS	5	64-32-16	0.4882	0.8136	0.7231	0.0185	0.8863
5/NFS	32	64-64-48-32-16	0.4504	0.8109	0.7232	0.0051	0.9567

[5]의 실험은 CNN이 MLP보다 훨씬 좋은 결과를 보였으므로 CNN을 구현하였다. 하지만 그들의 실험과 유사한 파라미터로 진행한 JM1에 대한 실험 결과는 표 7의 첫 번째 결과로 DNN보다 좋지 않은 결과를 보였다. 여러 파라미터를 변경해본 결과, 성능 향상을 위하여 컨볼루션 층을 한층씩으로 단순화하고, 에포크 수를 2000, 배치사이즈를 500, 출력층 드롭아웃을 0.5로 하였을 때 표 7의 두 번째 결과와 같은 더 나은 결과를 얻을 수 있었다. 하지만 이 결과도 DNN보다 좋지 않은 결과이다. 이는 CNN이 결함 예측 모델의 분류기 모델로 적합하지 않을 것이라는 예상과 일치한다. 모델 실험들은 모두 10 폴드 교차 검증(10-fold cross validation)을 사용하였고, CNN의 경우는 5 폴드의 경우도 실험하였다.

표 7. CNN 사용 실험결과
Table 7. Experimental results of CNN

에포크수/배치사이즈	폴드수	ACC	AUC
4,000/100	5	0.8006	0.632
	10	0.8068	0.584
2,000/500	5	0.8051	0.747
	10	0.7632	0.720

4. 이상치 데이터 제거 실험

모델의 성능을 높이기 위하여 이상치 데이터를 제거한 데이터 집합을 사용해보았다. 이상치 데이터 제거 작업은 논리적 또는 의미적으로 문제되는 모듈 데이터를 제거하는 것이 아니라 딥러닝에서 일반적으로 사용하는 이상치 제거 방법을 사용하였다. 이상치 데이터는 전체 데이터의 패턴에서 매우 심하게 벗어난 아웃라이어 데이터를 뜻한다. 이를 제거하는 방법은 여러 가지가 있지만 일반적인 IQR(Inter Quantile Range) 방법을 사용하였다.

차원축소 작업에서 입력 메트릭들 중 모델의 출력인 결함경향성과 가장 상관관계가 높은 결과가 나온 loc만을 이상치 데이터를 찾아내는데 사용하였다. 그리고 결함경향 데이터 집합에서는 결함경향 모듈이 비결함경향 모듈보다 현저히 적으므로 이상치 제거는 비결함경향 데이터만을 제거하는 것으로 하였다. 이와 같은 기법으로 779개의 이상치 모듈을 검출하여 제거하였고, 총 10,101개의 모듈 데이터 집합을 사용하여 추가 실험을 진행하였다. 파라미터 셋팅이 좋은 결과를 보인 모델들에 대해서만 해당 실험을 실시하였으며 표 8에서 보듯이 모든 성능 척도 결과가 좋아졌으며 특히 AUC와 FNR이 크게 좋아진 것을 알 수 있다.

표 8. 이상치 데이터 제거 실험 결과
Table 8. Experimental results when outlier data is removed

은닉층수	배치사이즈	노드수 변화	Loss	ACC	AUC	FPR	FNR
3	5	32	0.4482	0.8135	0.7702	0.0001	0.8749
		48	0.4397	0.8235	0.7724	0.0003	0.8458
		64	0.4404	0.8173	0.7717	0.0001	0.8768
		64-32-16	0.4401	0.8163	0.7700	0.0001	0.8815
5	5	64-64-48-32-16	0.4416	0.8202	0.7723	0.0001	0.8625

V. 결 론

수십년 간 수많은 소프트웨어 결함 예측 모델 연구들이 수행되었으며, 기계학습 기법들이 모델 구축에 많이 이용되었다. 최근 딥러닝 알고리즘이 매우 각광을 받고 있으나 딥러닝을 이용한 예측 모델 연구들은 대부분 소스 코드로 부터 시맨틱 피쳐 정보를 생성하는 분야에 사용되고 있으며 딥러닝 모델을 모델 분류기로 사용한 연구는 극소수에 불과하다.

본 연구는 3개 이상의 은닉층을 갖는 심층 신경망 모델들을 구현하여 기존 예측 모델들인 하나의 은닉층을 갖는 MLP(MLPs), 결정 트리 모델의 성능과 비교하였다. 심층 신경망 모델은 여러 하이퍼 파라미터 값과 은닉층의 수, 각 층의 노드수를 변화시키면서 여러 형태의 모델들을 구현하였다. 실험 결과 Accuracy는 큰 향상이 없었지만 AUC는 유의미한 성능 향상이 있었다. IQR 기법을 사용하여 이상치 데이터를 제거한 실험에서는 AUC와 FNR 결과가 탁월하게 좋아졌다. 또 다른 딥러닝 신경망인 CNN을 구현하여 제안 모델들과 비교하였지만 CNN의 성능은 좋지 않은 결과를 보였다.

References

[1] E. Hong, “Software Fault Prediction using Semi-supervised Learning Methods,” Journal of the Institute of Internet, Broadcasting and Communication, vol. 19, no. 3, pp. 127-133, 2019.
DOI: https://doi.org/10.7236/JIIBC.2019.19.3.127

[2] Z. Xu, L. Li, et al., "A comprehensive comparative study of clustering-based unsupervised defect prediction models", Journal of Systems and Software, vol. 172, pp. 110862, 2021.
DOI: https://doi.org/10.1016/j.jss.2020.110862

[3] S. Omri and C. Sinz, “Deep Learning for Software Defect Prediction: A Survey,” Proc. of ICSEW, pp. 209-214, July 2020.
DOI: https://doi.org/10.1145/3387940.3391463

[4] E. N. Akimova et al., "A survey on software defect prediction using deep learning", Mathematics, vol. 9, no. 11, 2021.
DOI: https://10.3390/math9111180

[5] O. A. Qasem, M. Akour and M. Alenezi, "The Influence of Deep Learning Algorithms Factors in Software Fault Prediction," IEEE Access, vol. 8, pp. 63945-63960, 2020.
DOI: https://10.1109/ACCESS.2020.2985290

[6] http://promise.site.uottawa.ca/SERepository

[7] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," arXiv preprint arXiv:1804.07612, 2018.
DOI: https://doi.org/10.48550/arXiv.1804.07612

[8] E. Hong, "Ambiguity Analysis of Defectiveness in NASA MDP data sets," Journal of Information Technology Services, vol. 12, no. 2, pp. 361-371, 2013.
DOI: https://doi.org/10.9716/KITS.2013.12.2.361

[9] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861-874, June 2006.
DOI: https://doi.org/10.1016/j.patrec.2005.10.010

[10] E. Frank, M. A. Hall, and I. H. Witten, The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[11] G. Yoon, "The Analysis of Software Quality Management of Weapon System in Development Phase," Journal of the Korea Academia-Industrial cooperation Society, Vol. 22, No. 10, pp. 117-123, 2021.
DOI: https://doi.org/10.5762/KAIS.2021.22.10.117

저 자 소 개

홍 의 석(정회원)

- 1992년 : 서울대학교 계산통계학과 전산과학전공 학사
- 1994년 : 서울대학교 계산통계학과 전산과학전공 석사
- 1999년 : 서울대학교 계산통계학과 전산과학전공 박사
- 현재 : 성신여자대학교 컴퓨터공학과 교수

- 관심분야 : 소프트웨어 품질 예측, AI 엔지니어링, MSR 등

※ 이 논문은 2019년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음.