이 일 구 교수 지도
석사학위 청구논문

악성 행위 경량 탐지를 위한 동적 데이터 축소 기법 선택 알고리즘

2024

성신여자대학교 대학원
미래융합기술공학과
김 민 정

악성 행위 경량 탐지를 위한 동적 데이터 축소 기법 선택 알고리즘

이 일 구 교수 지도

이 논문을 석사학위논문으로 제출함

2023년 11월

성신여자대학교 대학원

미래융합기술공학과

김 민 정

인 준 서

김민정의 석사학위 논문으로 인준함

2023년 11월

심사위원장 김 성 민 (인)

심 사 위 원 이 일 구 (인)

심 사 위 원 임 연 섭 (인)

성신여자대학교 대학원

논 문 개 요

최근 악성코드가 점차 다양해지고 공격 방식이 정교해지면서 인공지능을 기반으로 한 악성 행위 탐지 기법에 대한 연구가 활발하게 진행되고 있다. 정확한 악성 행위 탐지를 위해서는 인공지능 학습 및 추론 과정에서 대량의 데이터가 요구된다. 그러나 실시간으로 수집되는 빅데이터를 저장하고 학습하려면 고성능 하드웨어 및 고용량 메모리가 많이 필요하다는 한계점이 존재한다. 특히 IoT(Internet of Things)와 같이 사용 가능한 자원이 제약된 환경에서는 일반적인 악성 행위 탐지 기법을 적용하기 어려우므로, 정확도와 메모리 사용량, latency의 트레이드오프(Trade-off) 문제가 발생한다.

본 논문에서는 빅데이터 환경에서 메모리 비효율적인 악성 행위 탐지 문제를 해결하기 위해 Feature selection, Bit scaling, Undersampling 기법을 조합하여 최적의 데이터 축소 기법을 제안한다. 또한 컴퓨팅 환경에 맞게 데이터 축소 기법을 동적으로 선택하는 알고리즘을 제안하여 정확도 대비 메모리 효율성을 개선하였다. 시뮬레이션 결과에 따르면 종래 방법 대비 메모리 효율성을 약 90배의 개선하였으며, 무작위하게 생성한 환경 조건에서도 메모리 효율성을 약 7.6배 개선하였음을 증명했다. 따라서 제안하는 알고리즘은 기존의 방대한 데이터를 저장 및 학습하는 과정에서 발생하는 메모리 사용량 및 latency를 줄여서 악성 행위의 경량 탐지가 가능함을 보였다.

목 차

표 차 례

그림 차례

Ⅰ. 서 론

최근 사물인터넷, 클라우드 컴퓨팅(Cloud computing), 빅데이터(Big data) 등 4차 산업혁명을 주도하는 신기술들이 빠르게 발전함에 따라 생성되는 데이터의 양이 기하급수적으로 증가하고 있다[1]. 통계에 따르면 2020년에 전 세계적으로 64.2 제타바이트(ZB) 규모의 데이터가 생성 및 소비되었으며, 2025년에는 180 제타바이트까지 증가할 것으로 예상하고 있다[2].

증가하는 데이터는 인공지능(AI, Artificial Intelligence)의 학습 및 예측에 필요한 핵심 자원이 되었다. 특히 최근에 이러한 데이터를 기반으로 악성 행위를 탐지하고 대응하기 위한 인공지능과 관련된 연구들이 활발하게 진행되고 있다. 그러나 2019년 위협 보고서에 따르면[3], 2018년에는 약 9억 7,300만 개의 악성코드가 탐지되었으며 악성코드가 더욱 정교해짐에 따라[4], 인공지능 기반의 악성 행위 탐지 시스템이 정확한 탐지를 위해 학습에 필요한 데이터 및 자원이 증가하였다. 이는 데이터를 저장하고 학습하기 위해서 고성능 하드웨어 및 고용량 메모리가 요구된다는 문제를 초래했다. 또한 실시간으로 데이터를 처리하고 대응하는 능력이 중요한 악성 행위 탐지 시스템은[5], 대량의 데이터를 학습데이터로 사용하게 될 경우, 많은 학습 시간이 소요되기 때문에 실시간 탐지 및 대응이 어렵다. 특히 IoT 환경과 같이 컴퓨팅 파워와 메모리 용량이 제한된 환경에서는 일반적인 악성 행위 탐지 기법을 적용하기 어려우므로 자원이 제한된 환경에서도 빠르고 정확하게 악성 행위를 탐지하기 위한 경량 악성 행위 탐지 알고리즘이 필요하다.

따라서 본 논문에서는 악성 행위 탐지에 Bit scaling, Feature selection, Undersampling의 세 가지 데이터 축소 기법을 적용하여 메모리 효율적인 경량 악성 행위 탐지를 위한 새로운 동적 데이터 축소 기법 선택 알고리즘을 제안한다.

본 논문의 주요 기여점은 다음과 같다.

1) 악성 행위 경량 탐지 및 메모리 효율적인 데이터 사용을 위한 동적 데이터 축소 기법 선택 알고리즘을 제안한다.

2) 제안하는 기법은 메모리 사용이 제한적인 상황에서 최적의 데이터 축소 기법을 동적으로 선택한다.

3) 제안하는 알고리즘은 정확도, 메모리 사용량, latency 간 트레이드오프 문제를 해결하여 메모리 사용량 및 latency의 효율성을 최대화할 수 있다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 데이터 축소기법 및 악성 행위 탐지와 관련된 연구를 소개하고, Ⅲ장에서는 데이터 축소 기법을 설명한다. Ⅳ장에서는 메모리 효율적인 경량 악성 행위 탐지를 위한 동적 데이터 축소 기법 알고리즘을 제안하고 Ⅴ장에서는 제안한 알고리즘의 성능을 평가하며 마지막 Ⅵ장에서는 본 연구를 요약하고 마무리한다.

Ⅱ. 선행 연구

본 장에서는 Feature selection, Bit scaling, Undersampling 세 가지 데이터 축소 기법의 선행 연구를 분석한다.

TABLE Ⅰ은 Feature selection 기법과 관련된 선행 연구에서 사용한 데이터셋과 제안한 기법을 요약한 표이다. Feature selection 기법을 활용해 feature의 수를 줄여서 악성 행위 탐지 정확도를 개선하거나, 메모리를 감소시키는 연구가 주를 이루고 있다.

TABLE Ⅰ

Previous studies on feature selection

Ref.	Dataset	Methods
[6]	EDBDL'14	• GBT(Gradient Boosted Trees), LR(Logistic Regression), RF(Random Forest) 모델과 다른 크기의 데이터셋, 다양한 비율 조합을 활용하여 모델을 생성하고, 성능을 분석
[7]	KDDCUP'99, UNSW-NB15	• KDD'99 데이터셋에서 feature를 선별하여 새로운 데이터셋인 NSL-KDD를 제안하여 성능을 향상
[8]	Malware Detection, Android Malware Dataset for Machine Learning	• 각각의 데이터셋에서 feature를 줄여서 성능을 평가하였고, feature selection의 효과를 입증
[9]	CCCS-CIC-AndMal-2020	• Feature selection 기법과 누락 데이터 대치, Random oversampling, 이상값 처리, 원핫 인코딩 등 기법 간의 앙상블을 통해 feature를 줄이고 정확도 향상

Tawfiq Hasanin [6]은 GBT(Gradient Boosted Trees), LR(Logistic Regression), RF(Random Forest) 3개의 모델과 다른 크기의 3가지 데이터셋, 6개의 비율을 활용하여 총 54개의 모델을 생성하고, 모든 조합의 성능을 분석하여 효과적인 방법을 도출하였다. 하지만 성능 평가에서 선별된 feature들을 그대로 사용하였으며, 실험에 feature selection 기능을 포함하지 않았다.

Tharmini Janarthananl [7]은 KDD'99 데이터셋에서 feature를 선별하여 새로운 데이터셋인 NSL-KDD를 제안하였다. 해당 연구에서 데이터셋의 크기를 줄이고 침입 탐지 성능을 개선했지만 KDD'99 데이터셋 이외의 다른 데이터셋에 대한 성능을 보장할 수 없었다.

Esraa Saleh Alomari [8]은 각각의 데이터셋에 feature selection 기법을 적용하여 감소한 데이터셋의 성능을 평가하였다. 두 데이터셋에 대해 데이터 사이즈가 각각 최대 42.42%, 93.5% 감소한 것에 비해 정확도는 최대 5.84%, 9.44% 저하되어 feature selection의 효과를 입증하였다. 하지만 데이터의 사이즈가 감소함에 따른 성능 저하 문제를 해결하지 못하였다는 점에서 성능과 자원과의 트레이드오프 문제가 있다.

Rejwana Islam [9]는 feature selection 기법과 누락 데이터 대치, 랜덤 오버샘플링, 이상값 처리, 원핫인코딩 기법 사이의 앙상블 기법을 제안하여 feature를 60.2% 줄이고 정확도를 0.8% 개선할 수 있었다. 하지만 성능 평가에 latency를 고려하지 않았으며 데이터 변화에 따른 성능을 평가하지 않았다는 한계점이 있다.

TABLE Ⅱ는 Bit scaling 기법에 관한 선행 연구를 요약한 표이다.

TABLE Ⅱ

Previous studies on bit scaling

Ref.	Dataset	Methods
[10]	ImageNet	• 양자화 프레임워크를 제안하여 처리 속도 및 에너지 효율성을 개선하고 메모리 비용을 감소 • 모든 작업을 비트로 대체할 수 있어서 메모리 오버헤드, 처리 속도 및 에너지 소비를 개선
[11]	ImageNet	• 8비트 부동 소수점을 활용하여 데이터 및 계산 정밀도를 8bit로 줄였으며 정확도 저하 없이 최소 2배의 속도 향상
[12]	RT-IoT2022	• QAE(Quantized Autoencoder) 모델과 QAE를 변형한 QAE-u8, QAE-f16 모델을 제안 • 이상 징후를 탐지하기 위한 침입 탐지 시스템 성능을 평가하여 제안한 QAE-u8 모델이 리소스가 제한된 IoT 환경에 더 적합함을 증명
[13]	CIFAR-10, ImageNet, IWSLT'15 English-Vietnamese Dataset, MovieLens 1 Million Dataset	• 8비트 부동 소수점을 이용한 S2FP8을 제안하여 비트 정밀도를 감소시키고 효과적인 메모리 사용 및 향상된 연산을 가능하게 함

Yukuan Yang [10]은 양자화 프레임워크를 제안하여 처리 속도 및 에너지 소비 효율을 개선하고 메모리 비용을 감소시켰다. 제안한 양자화 방식은 모든 작업을 비트로 대체할 수 있어서 메모리 오버헤드, 처리 속도 및 에너지 소비를 크게 향상할 수 있었다. 하지만 메모리 오버헤드, 처리 속도, 메모리 비용만 평가하고, 정확도와 메모리 효율성의 상관관계를 분석하지 않았다는 한계가 있다.

Naigang Wang [11]은 8비트 부동 소수점을 활용하여 데이터 및 계산 정밀도를 8비트로 줄였으며 정확도 저하 없이 최소 2배 이상의 속도를 개선했다. 하지만 데이터셋을 imagenet에 한정하여 다른 데이터셋에서도 비슷한 성능을 유지하는지 알기 어렵다는 한계점이 있다.

B. S. Sharmila [12]은 QAE(Quantized Autoencoder) 모델과 QAE를 변형한 QAE-u8, QAE-f16 모델을 제안했다. 그리고 이상 행위를 탐지하기 위한 침입 탐지 시스템 성능을 평가하여 메모리 크기, CPU 사용량 측면에서 성능이 향상됨을 입증하였다. 하지만 Raspberry Pi IoT 장치에서 autoenocder, QAE-f16 및 QAE-u8 모델의 성능 평가에서는 정확도를 평가하지 않았고 메모리 사용량 분석에만 집중되었다는 한계점이 있다.

Leopold Cambier [13]은 8비트 부동 소수점 형식을 활용한 S2FP8(Shifted and Squeeze FP8)을 제안했다. S2FP8은 비트 정밀도를 감소시켜 양자화로 인한 정보 손실을 최소화하였으며 빠른 훈련이 가능함을 보였다. 하지만 BLUE(Bilingual Evaluation Understudy)와 NDCG(Normalized Discounted Cumulative Gain)를 평가 지표로 사용하였기 때문에 다른 데이터셋에 대한 일반화 성능을 보장할 수 없었다.

TABLE Ⅲ은 Undersampling 기법에 관한 선행 연구를 정리한 표이며, 주로 불균형 데이터셋에서 다양한 Undersampling 기법을 통해 데이터의 균형을 맞추고 정확도를 개선하는 연구가 주를 이루었다.

TABLE Ⅲ

Previous studies on undersampling

Ref.	Dataset	Methods
[14]	nps-2009-casper-rw, Honeynet Forensic Challenge 7	• Oversampling 기법 및 Undersampling 기법을 적용하여 데이터 균형이 이상 탐지의 성능을 향상시킴을 증명
[15]	SGCC dataset	• 불균형 시계열 데이터에 RUS (Random Undersampling) 기법을 적용하여 다양한 ML(Machine Learning) 알고리즘에서의 성능 비교 • RUS 기법이 과소 적합(underfitting)을 유발하지 않았음을 증명
[16]	CSE-CIC-IDS2018 , NSL-KDD	• RL(Reinforcement learning)을 기반으로 설계된 RLFOUA(Oversampling and Undersampling Algorithms) 프레임워크를 제안하고 TFRSMOTE(True False Rate Synthetic Minority Oversampling Technique) 알고리즘을 통합하여 AESMOTE에 비해 정확도가 향상됨을 보임
[17]	CSE-CIC-IDS2018	• RUS(Random Undersampling) 비율을 변화시켜 데이터셋의 불균형 문제를 해결하였고 RUS 기법의 유효성을 증명

Hudan Studiawan [14]은 불균형 데이터셋에 4가지 Oversampling 기법 (Random Oversampling, Synthetic Minority Oversampling Technique (SMOTE), Adaptive Synthetic (ADASYN), SVM Combined with SMOTE (SVM-SMOTE)), 4가지 Undersampling 기법 (Random Undersampling, Tomek Links, Near-miss, Instance Hardness)을 적용할 때 균형 데이터를 활용하면 이상 탐지 성능을 개선할 수 있음을 보였다. 하지만 메모리 사용량, latency 등 다른 평가 지표를 고려하지 않았기 때문에 자원이 제한된 환경에 적용하기에는 어려움이 있다.

Mulyana Saripuddin, Ms [15]은 불균형 시계열 데이터에 RUS(Random Undersampling) 기법을 적용하고 다양한 ML(Machine Learning) 알고리즘에서의 성능을 비교하여 RUS 기법이 과소 적합을 유발하지 않았음을 증명하였다. 하지만 시계열 데이터에 한정되어 있으며, undersampling 비율을 60:40으로 설정했을 때를 제외하고는 정확도가 향상되지 않았으므로 일관된 결과를 보인다고 판단하기에는 어려움이 있다.

Najmeh Abedzadeh [16]은 RL(Reinforcement learning)을 기반으로 설계된 RLFOUA(Oversampling and Undersampling Algorithms) 프레임워크를 제안하고 TFRSMOTE (True Falsse Rate Synthetic Minority Oversampling Technique) 알고리즘을 통합하여 기존 리샘플링 기법인 AESMOTE (Adversarial Reinforcement Learning with SMOTE for Anomaly Detection)에 비해 정확도가 21.5% 향상됨을 보였다. 하지만 RLFOUA 프레임워크의 성능을 비교한 AESMOTE 프레임워크의 신뢰성이 보장되지 않는다는 한계점이 있다.

Richard Zuech [17]은 RUS(Random Undersampling) 비율을 달리하여 데이터셋의 불균형 문제를 해결하였고 RUS 기법의 유효성을 증명하였지만 메모리 사용량, latency 등을 고려하지 않았다.

Ⅲ. 데이터 축소 기법

1. Feature selection

Feature selection은 데이터 학습 모델 성능에 영향력이 큰 feature를 선택하여 데이터를 축소하는 기법이다. 이를 통해 학습 모델의 복잡도와 비용을 줄이고, 과적합을 방지하면서 학습 속도를 개선한다. Feature selection은 Filter 기법, Wrapper 기법, Embedded 기법으로 나뉜다.

Filter 기법은 통계적 속성에 기반하여 대표적인 feature를 선택하는 방법이다. Feature가 학습하려는 목표 변수를 얼마나 잘 설명하는지 측정하여 feature를 선택하거나, 분산이 작은 경우에는 학습에 유용하지 않을 가능성이 크다고 판단하여 feature를 제거한다[18]. 이 기법은 모델을 학습하는 과정이 아닌 데이터 전처리 단계에서 feature를 선택하기 때문에 학습 모델의 성능에 의존하지 않고 빠르게 feature를 선택할 수 있다.

Wrapper 기법은 학습 모델의 성능을 고려하여 전체 feature 중 feature 부분 집합의 중요도를 평가하는 방식이다. 학습 모델 성능이 향상되는 feature를 하나씩 추가하거나, 성능이 떨어지는 feature를 제거하여 최종 feature 집합을 선택한다[19]. 이 기법은 모든 feature의 성능을 평가하기 때문에 feature 선택에 필요한 비용이 증가할 수 있지만, 정확한 feature를 선택할 수 있다.

마지막으로 Embedded 기법은 학습 모델이 데이터를 학습하는 과정에서 학습 모델의 성능 향상에 기여하는 feature를 선택하거나 가중치를 조절하여 feature를 선택하는 방법이다[20]. 이 기법은 모델 학습 중에 feature 선택이 이루어지기 때문에 학습 모델에 의존한다는 한계가 있지만, 별도의 선정 기준이 없어도 성능 향상에 영향을 주는 feature를 선택한다.

2. Bit scaling

Bit scaling은 실수 데이터를 정수 데이터로 변환하는 기법으로, 데이터의 표현 정밀도를 낮추고 데이터의 크기를 줄여서 메모리 사용량을 절약할 수 있다. Bit scaling 기법은 두 단계를 거쳐 진행된다. 첫 번째는 부동 소수점 값을 정수로 변환하는 단계이다. 실수 데이터를 정수로 변환할 때 소수점 이하를 버려서 정수 부분만 남기므로 데이터의 정밀도를 감소시키는 효과가 있다. 두 번째는 이전 단계에서 정수로 변환된 데이터를 스케일링할 bit 값으로 나누거나 곱하는 스케일링 단계이다.

TABLE Ⅳ는 Bit scaling 기법을 적용할 때 표현할 수 있는 값을 나타낸 표이다. 16bit scaling과 8bit scaling은 각각 0~65,535, 0~255까지의 값을 표현할 수 있다. 4bit scaling, 2bit scaling, 1bit scaling은 데이터셋을 8bit로 변환하고 하위 4bit, 2bit, 1bit만을 사용하기 때문에 4bit scaling은 0~15, 2bit scaling은 0~3, 1bit scaling은 0~1의 값을 표현할 수 있다.

TABLE Ⅳ

Expressible values by bit

Scaling bit	Expressible value
1bit	0~1
2bit	0~3
4bit	0~15
8bit	0~255
16bit	0~65,535

이는 데이터를 일정 범위 내로 압축하여 저장 공간을 절약하고 연산에 필요한 비용이 감소하는 효과가 있다. 하지만 Bit scaling은 소수점 이하의

정보를 삭제하기 때문에 정밀도 손실이 발생할 수 있으므로 적절한 bit 값을 선택하는 것이 중요하다.

3. Undersampling

Undersampling은 불균형 데이터셋에서 클래스 간 샘플 수의 차이가 클 때, 상대적으로 많은 샘플이 속한 클래스의 데이터를 줄여서 데이터 불균형 문제를 해결하기 위한 기법이다. 데이터셋을 축소하기 때문에 데이터 손실이 발생할 수 있지만, 학습 속도 및 메모리 사용량 측면에서 효율성이 개선되고 성능이 향상되는 장점이 있다. Undersampling 기법은 Random undersampling, Tomek links, Near-miss, Instance Hardness, Undersampling 기법으로 나눌 수 있다.

Random undersampling은 가장 단순하고 기본적인 방법으로, 불균형 데이터셋 중 비교적 많은 데이터를 가진 클래스에서 제거할 데이터를 무작위로 선택한다[21]. 이때 무작위성을 제거하기 위해서 무작위 시드(seed) 값을 설정하기도 한다. Random undersampling을 수식으로 나타내면 수식 (1)과 같이 표현할 수 있다.

$$M_{undersampled} = Random\,Selection(M,N) \tag{1}$$

Table V와 같이 수식 기호에 대한 표기법을 정의할 때, Random undersampling은 M에서 N개의 샘플을 무작위로 제거하여 Undersampling을 수행한다.

TABLE V

Notation and Description of numerical analysis

Notation	Description
M	The majority class of unbalanced dataset
N	The minority class of unbalanced dataset

Tomek links는 서로 다른 클래스에 속하는 가장 가까운 이웃 데이터 쌍인 Tomek pair를 찾고, 상대적으로 많은 샘플이 속한 클래스에 해당하는 데이터를 제거하거나 이웃 쌍을 제거하여 두 클래스 간의 경계를 명확하게 만드는 기법이다[22]. Tomek links를 그림으로 표현하면 FIGURE 1과 같이 나타낼 수 있다.

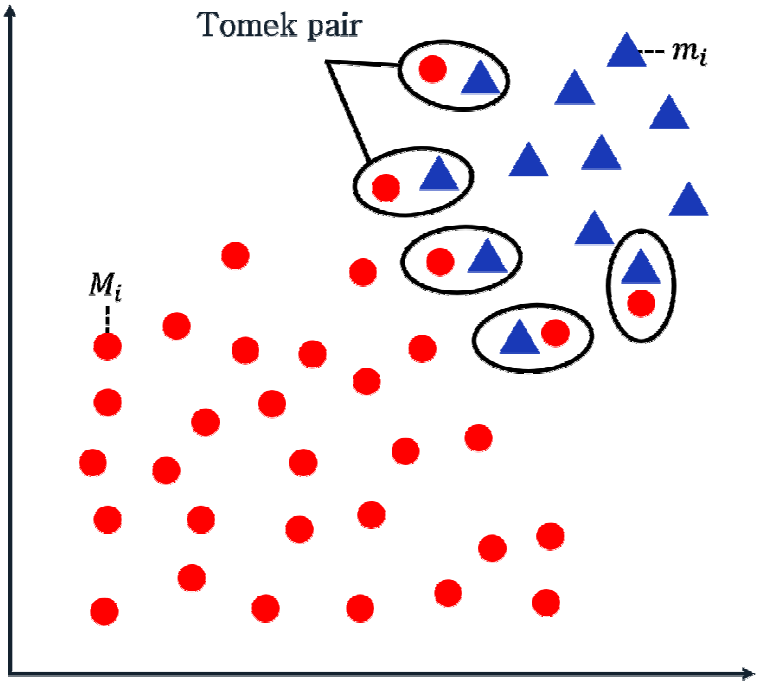

FIGURE 1. Description of tomek links

M에 속하는 데이터를 M_i, m에 속하는 데이터를 m_i라고 표현할 때 Tomek links는 전체 데이터에서 Tomek pair를 제거한 차집합이다.

Near-miss는 M과 m사이의 거리를 고려하여 m의 일부를 제거하는 방식이다[23]. Near-miss 버전은 샘플을 선택하거나 제거하는 방법에 따라 NearMiss-1, NearMiss-2, NearMiss-3으로 분류한다. Near-miss를 수식으로 나타내면 수식 (2)와 같이 표현할 수 있다.

$$M_{undersampled} = \{x_i \mid x_i \in m, \text{ and } NM1,2,3(x_i, M)\} \quad (2)$$

x_i는 데이터 하나를 의미하고 NM1, NM2, NM3은 Near-miss 버전을 의미할 때, Near-miss 기법은 M에 해당하면서 Near-miss 버전에 따라 샘플을 선택하거나 제거한다.

Instance Hardness는 데이터셋에서 각각의 인스턴스가 얼마나 어려운지, 즉 얼마나 정확히 분류하기 어려운지 계산하고 해당 데이터를 제거하여 불균형을 해소하는 방법이다[24].

Ⅳ. 악성 행위 경량 탐지를 위한 동적 데이터 축소 기법

1. 동적 데이터 축소 기법 선택

종래 연구는 악성 행위 탐지 정확도 향상에 초점을 맞춰 AI 모델의 정확도 및 신뢰성을 높이는 데 중점을 두고 있다[25]. 그러나 높은 정확도를 위해서는 비교적 많은 양의 데이터와 충분한 학습 시간이 필요하므로 컴퓨팅 파워와 계산 능력이 비교적 낮고, 메모리 용량이 작은 환경에서는 일반적인 악성 행위 탐지 기법을 적용하기 어렵다. 따라서 정확도와 메모리 사용량, latency는 트레이드오프 관계에 있고 이들 사이의 균형을 찾아 최적화하는 연구가 필요하다[26].

본 절에서는 경량 악성 행위 탐지를 위한 동적 데이터 축소 기법을 선택하는 알고리즘을 제안한다. 정확도만 고려하거나 효율성에만 초점을 맞추었던 종래 연구의 한계점을 개선하기 위해서 학습 모델의 성능을 비롯하여 메모리 사용량과 latency를 고려하였다. 제안하는 알고리즘은 메모리 효율성을 평가한 후, 제한된 컴퓨팅 환경이나 요구사항에 맞게 데이터 축소 기법을 동적으로 선택한다.

본 논문에서 제안하는 알고리즘은 두 가지 이점이 있다.

첫 번째는 제한된 컴퓨팅 환경 및 요구사항을 모두 반영하면서 동시에 정확도 또한 고려하여 제한된 컴퓨팅 환경에서도 정확한 악성 행위 탐지가 가능하다.

두 번째는 정확도, 메모리 사용량, latency 간의 트레이드오프 문제를 해결하여 메모리 사용량, latency 효율을 최대화할 수 있다.

2. 동적 데이터 축소 기법 선택 알고리즘

동적 데이터 축소 기법 선택 알고리즘은 먼저 컴퓨팅 환경 요구사항이 있는지 판단하고, 요구사항이 없다면 가장 높은 정확도를 가진 기법을 선택한다. 요구사항 중 메모리 사용량 요구사항이 있다면 메모리 사용량 요구사항을 만족하는 기법 중 가장 높은 정확도를 갖는 기법을 선택하며 latency 요구사항이 있는 경우 이를 만족하는 기법 중 가장 높은 정확도를 갖는 기법을 선택한다. 앞서 설명한 알고리즘을 정리하면 FIGURE 2와 같다.

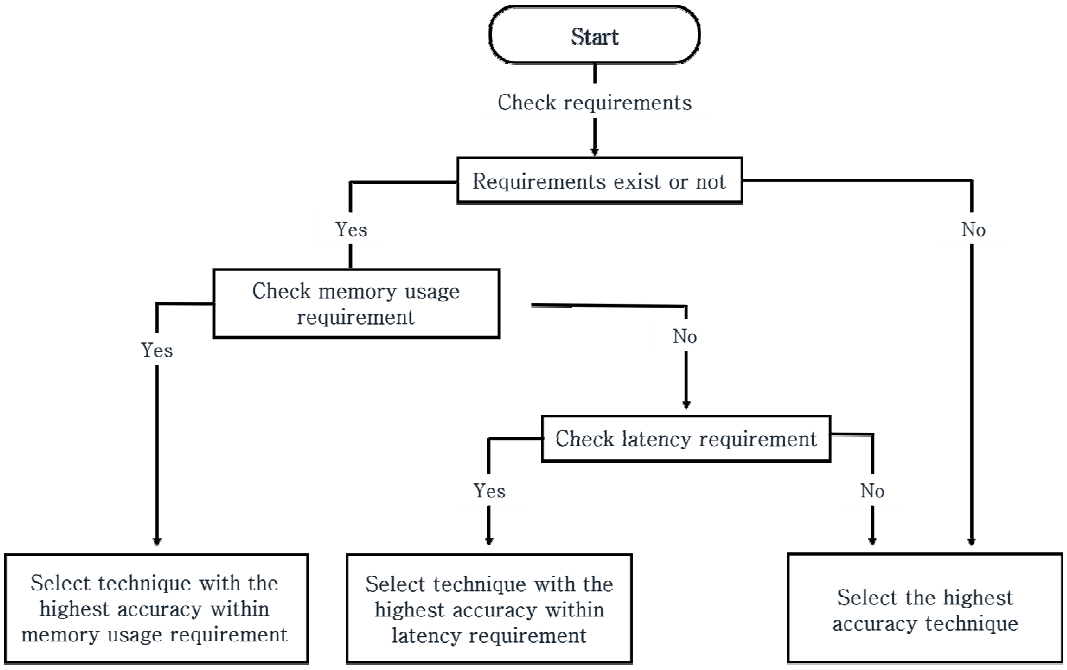

FIGURE 2. Proposed algorithm flowchart

Ⅴ. 성능 평가 및 분석

1. 평가 환경

본 장에서는 제안하는 알고리즘이 정확도와 메모리 사용량, latency 간의 트레이드오프 문제를 해결하고 제한된 컴퓨팅 환경에서도 최적의 데이터 축소 기법을 선택하여 악성 행위 탐지가 가능함을 입증하고자 한다.

평가에 사용한 데이터셋은 난독화된 악성코드 데이터셋인 CIC-Malmem-2022를 사용했다[27]. 총 58,596개의 데이터 중 악성코드 데이터는 29,298개, 정상 데이터는 29,298개로 각각 50%씩 구성되어 있다.

또한 데이터셋에서 다루고 있는 공격 유형은 랜섬웨어, 스파이웨어, 트로이목마 총 3가지이며 시뮬레이션을 위해 TABLE Ⅵ와 같이 Benign은 0, 랜섬웨어는 1, 스파이웨어는 2, 트로이목마는 3으로 라벨인코딩을 진행했다.

TABLE Ⅵ

Label encoding process

Attack Type	Label encoding
Benign	0
Ransomware	1
Spyware	2
Trojan Horse	3

TABLE Ⅶ은 CIC-Malmem-2022 데이터셋의 대표 feature 정보 및 feature에 대한 설명을 나타낸 표이며 CIC-Malmem-2022 데이터셋은 총 55개의 feature가 사용되었다.

TABLE Ⅶ

Representative features information of dataset

Features	Description
pslist	Process information
dlllist	DLL information
handles	Handles information
ldrmodules	Loader modules information
malfind	Malicious code detection information
psxview	Process extension information
modules	Modules information
svcscan	Service scan information
callbacks	Callback information

제안하는 방식과 종래 방식의 악성 행위 탐지 성능 비교를 위해 동일 환경 조건에서 정확도, 메모리 사용량, latency를 측정하였다. latency는 파이썬의 time함수를 사용하여 측정하였고 메모리 사용량은 각 데이터셋의 크기를 측정하였다. 정확도 및 성능 평가는 Scikit-learn 모듈에서 제공하는 accuracy_score 함수를 사용하였고 수식 (3)과 같이 나타낼 수 있다.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

TP (True Positive)는 실제 값이 양성인 경우에 이를 정확하게 분류한 경우이며, TN (True Negative)은 실제 값이 양성인 경우를 잘못 분류한 경우이다. FP (False Positive)는 실제 값이 음성(Negative)인 경우를 양성(Positive)으로 잘못 예측한 경우이고, FN (False Negative)는 실제 양성

(Positive)인 경우를 음성(Negative)으로 잘못 예측한 경우를 의미한다.

악성 행위 탐지 성능 평가는 FIGURE 3과 같은 과정으로 진행하였다. 전처리한 데이터셋을 40%의 훈련 데이터, 30%의 검증 데이터, 30%의 테스트 데이터로 각각 분할하고 Random Forest 모델을 사용하여 악성 행위 탐지 성능을 측정하였다. 이 과정을 1,000번 반복하여 통계적인 신뢰성을 확보하였다.

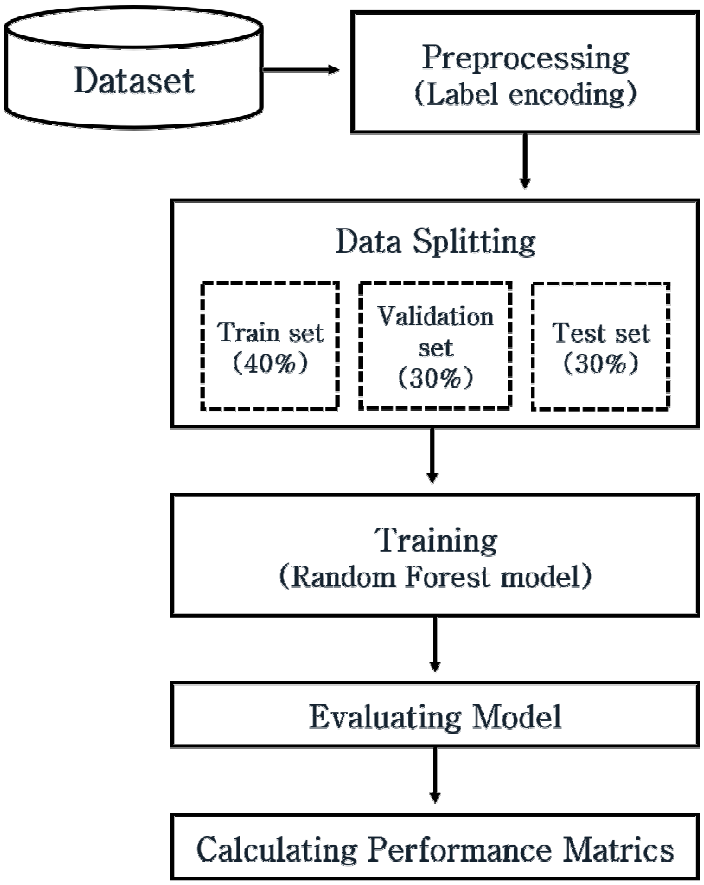

FIGURE 3. Flowchart of performance evaluation process

또한 제안하는 방식과 종래 방식의 성능을 비교하기 위한 지표로 메모리 효율성 지표를 활용하였다. 메모리 효율성은 정확도를 메모리 사용량과 latency의 곱으로 나눈 값이며 수식 (4)와 같이 나타낼 수 있다.

$$Efficiency = \frac{Accuracy}{Memory\ usage \times Latency} \tag{4}$$

2. 평가 결과 및 분석

제안하는 방식과 종래 방식의 성능 및 메모리 효율성을 비교하기 위해 총 두 가지 시뮬레이션을 진행하였다.

1) CIC-Malmem-2022 데이터셋 기반 시뮬레이션

첫 번째 시뮬레이션은 CIC-Malmem-2022 데이터셋에 각 데이터 축소 기법을 적용하여 종래 방식과 제안하는 방식의 메모리 효율성을 평가하였으며 시뮬레이션은 다음과 같은 과정으로 진행하였다.

CIC-Malmem-2022 데이터셋에 Feature selection, Bit scaling, Undersampling 기법을 적용하고, 최적의 메모리 효율을 보이는 파라미터 값을 각각 도출한다. 그리고 최적의 파라미터를 적용한 Feature selection 데이터셋, 최적의 파라미터를 적용한 Bit scaling 데이터셋, 최적의 파라미터를 적용한 Undersampling 데이터셋을 각각 생성하고, 이 3가지 데이터셋에 다른 데이터 축소 기법들을 적용하여 조합 가능한 모든 데이터셋을 생성한다. 다른 데이터 축소 기법들을 적용할 때 파라미터는 각 기법의 최적의 메모리 효율 지점을 나타내는 파라미터를 사용한다. 생성한 모든 데이터셋을 바탕으로 성능을 평가하고 종래 방식과 제안하는 방식의 메모리 효율성을 비교한다. 시뮬레이션 과정을 정리하면 FIGURE 4와 같다.

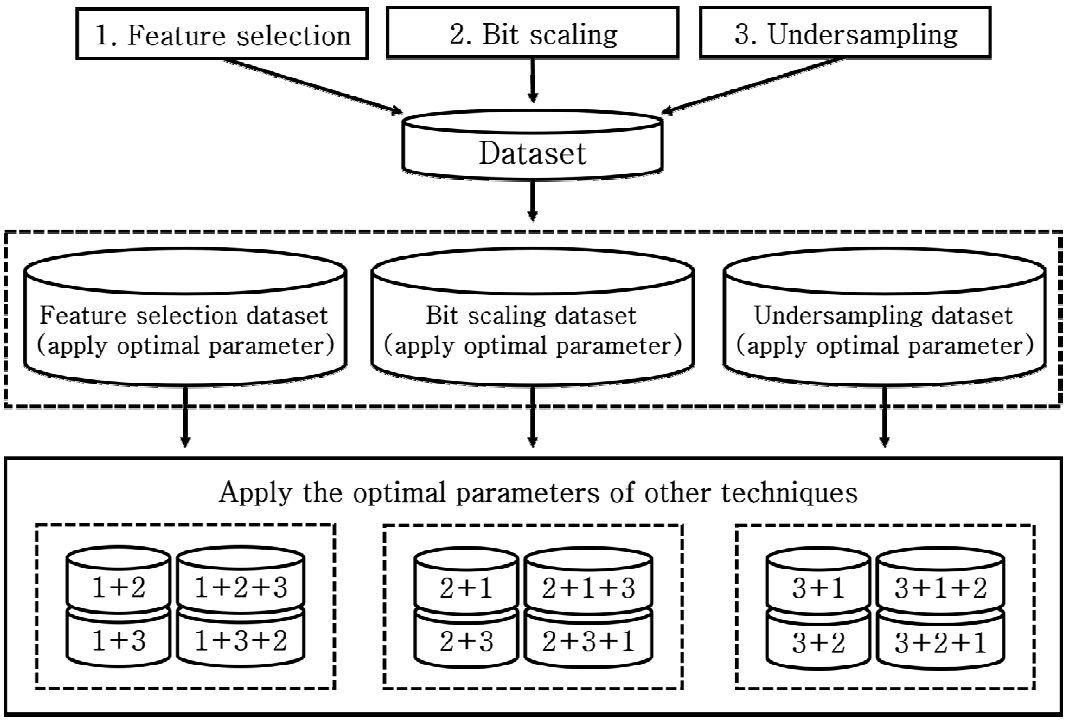

FIGURE 4. Flowchart of dataset utilization simulation process

Feature selection 기법을 적용할 때 최적의 메모리 효율 지점을 도출하기 위한 시뮬레이션 과정은 다음과 같다. 전처리한 데이터셋을 훈련 데이터, 검증 데이터, 테스트 데이터로 각각 40%, 30%, 30%씩 분할한다. Random Forest 모델을 사용하여 모델 성능 향상에 기여도가 높은 순서대로 feature를 선택하였고, feature 순서대로 1개부터 55개까지 feature 개수를 한 개씩 늘려가며 성능을 평가하였다.

FIGURE 5는 학습 모델 성능 향상에 기여도가 높은 feature 순서대로 정렬한 그래프이며 TABLE Ⅷ은 기여도가 높은 상위 15개의 feature와 각 feature에 대한 설명을 나타낸 표이다.

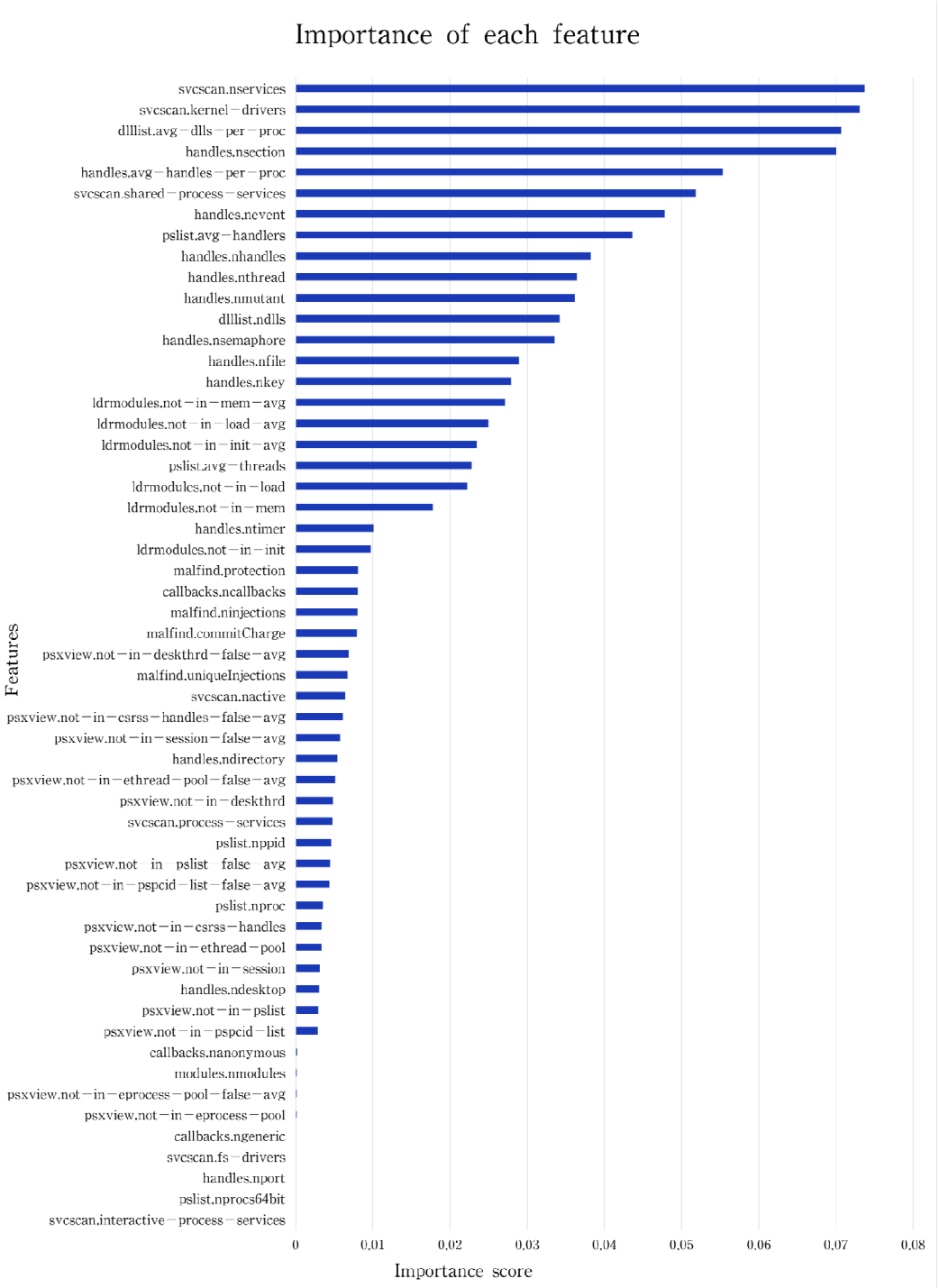

FIGURE 5. Importance of each feature

TABLE Ⅷ

Top 15 highly important features

Number	Features	Description
45	svcscan.nservices	Number of services
46	svcscan.kernel_drivers	Number of kernel drivers
6	dlllist.avg_dlls_per_proc	Average number of DLLs per process
18	handles.nsection	Number of memory sections per process
8	handles.avg_handles_per_proc	Average number of handles per process
49	svcscan.shared_process_services	Shared process services
11	handles.nevent	Number of events per process
4	pslist.avg_handlers	Average number of handlers per process
7	handles.nhandles	Number of handles per process
14	handles.nthread	Number of threads per process
19	handles.nmutant	Number of mutants per process
5	dlllist.ndlls	Number of loaded DLLs per process
16	handles.nsemaphore	Number of semaphores per process
10	handles.nfile	Number of files per process
13	handles.nkey	Number of registry keys per process

시뮬레이션 결과에 따르면, class 1인 랜섬웨어를 탐지할 때는 feature 2개를 사용할 때 최적의 메모리 효율성을 보였고, class 2인 스파이웨어와 class 3인 트로이목마를 탐지할 때는 feature 3개를 사용했을 때 가장 높은 메모리 효율성을 보였다. FIGURE 6은 각 class 별 메모리 효율성을 나타낸 그림이다.

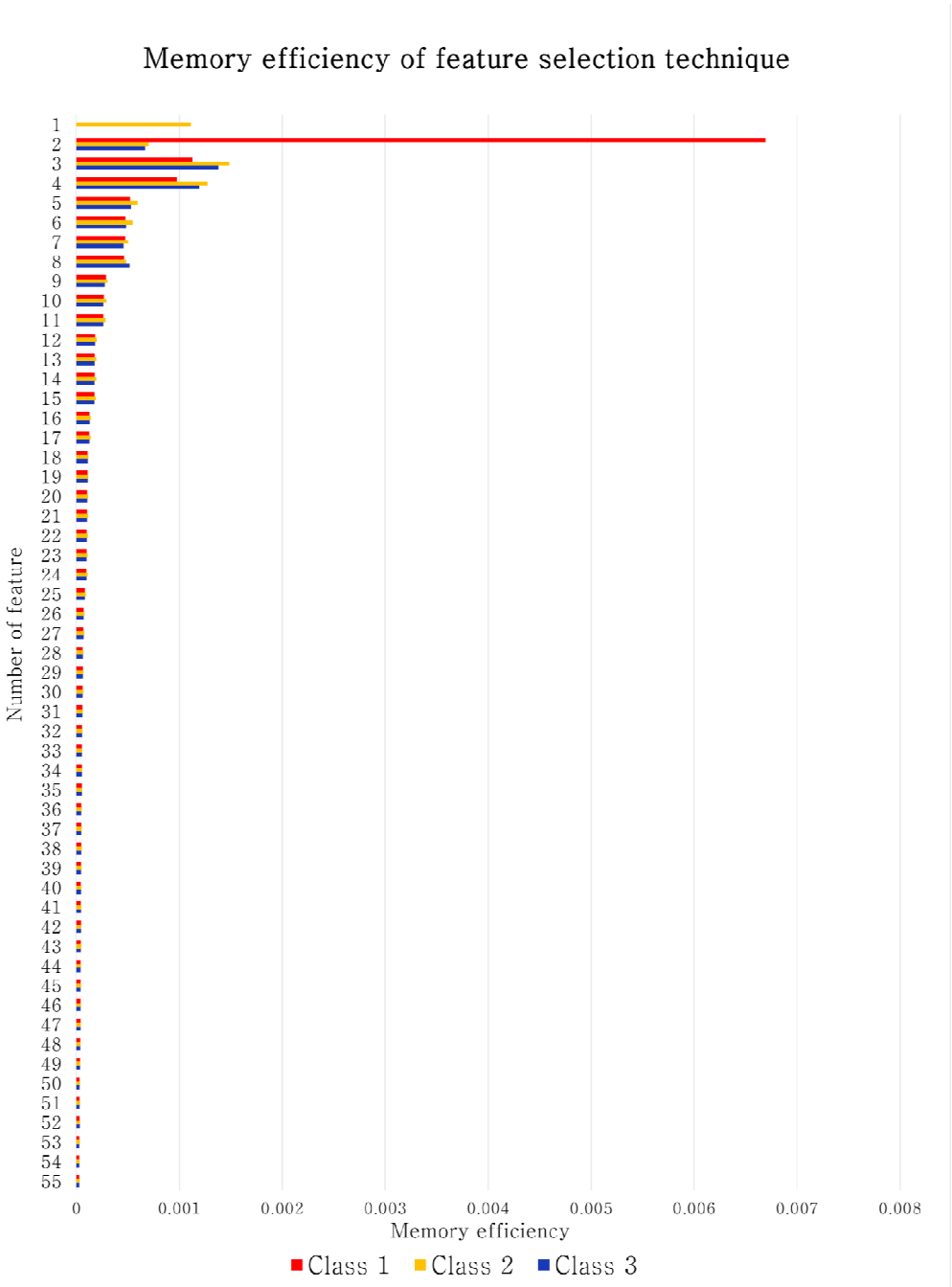

FIGURE 6. Memory efficiency of feature selection technique

Bit scaling 기법을 적용하였을 때 최적의 메모리 효율성을 나타내는 지점을 도출하기 위한 시뮬레이션 과정은 다음과 같다.

전처리한 데이터셋을 각각 1bit, 2bit, 4bit, 8bit, 16bit로 변환하여 새로운 데이터셋을 생성하고, 이 데이터셋을 바탕으로 악성 행위 탐지 성능을 평가하였다. TABLE Ⅸ~Ⅺ은 각 bit 별 정확도, 메모리 사용량, latency 성능 평가 결과를 나타낸 표이며 FIGURE 7은 Bit 크기에 따른 정확도 대비 메모리 효율성을 비교한 그래프이다. 시뮬레이션 결과에 따르면 메모리 사용량과 latency는 bit의 크기가 증가함에 따라 증가하였다. 또한 bit의 크기가 증가하면 많은 양의 데이터를 학습할 수 있으므로 정확도가 향상되는 양상을 보였지만 모든 class에서 2bit scaling을 적용하였을 때 가장 높은 메모리 효율성을 보였다.

TABLE Ⅸ

Bit scaling result of class 1

Performance metric	1bit	2bit	4bit	8bit	16bit
Accuracy (%)	58.45	67.38	70.49	70.88	71.27
Memory usage (kb)	6467.4	6467.4	7263.52	8996.33	9512.66
Latency (s)	100.06	112.26	128.49	118.6	129.73
Memory efficiency	9.03×10^{-15}	9.2×10^{-15}	7.5×10^{-15}	6.6×10^{-15}	5.7×10^{-15}

TABLE X

Bit scaling result of class 2

Performance metric	1bit	2bit	4bit	8bit	16bit
Accuracy (%)	63.0346	73.4725	74.5926	75.5091	76.5784
Memory usage (kb)	6467.4	6467.4	7263.52	8996.33	9512.66
Latency (s)	100.06	112.26	128.49	118.6	129.73
Memory efficiency	9.7×10^{-15}	10.1×10^{-15}	7.9×10^{-15}	7×10^{-15}	6.2×10^{-15}

TABLE XI

Bit scaling result of class 3

Performance metric	1bit	2bit	4bit	8bit	16bit
Accuracy (%)	56.9858	65.6724	70.7482	72.0041	71.9518
Memory usage (kb)	6467.4	6467.4	7263.52	8996.33	9512.66
Latency (s)	100.06	112.26	128.49	118.6	129.73
Memory efficiency	8.8×10^{-15}	9×10^{-15}	7.5×10^{-15}	6.7×10^{-15}	5.8×10^{-15}

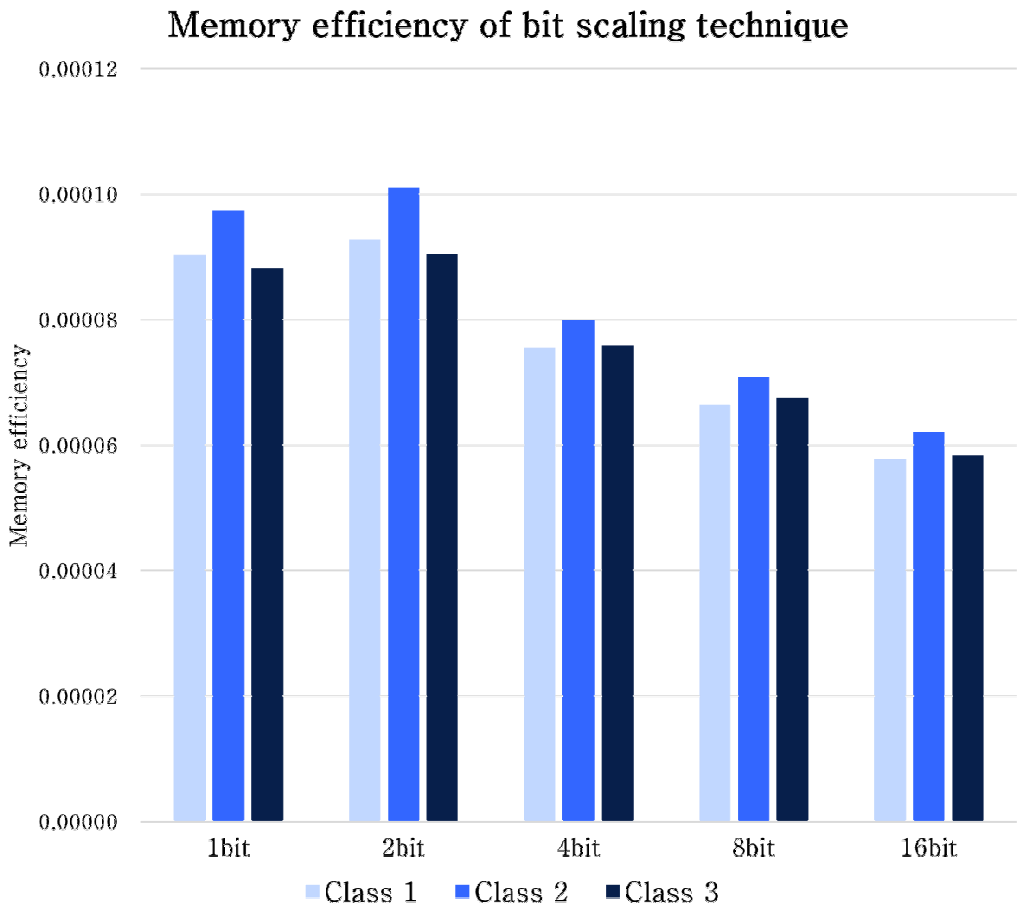

FIGURE 7. Memory efficiency of bit scaling technique

Undersampling 기법을 적용하였을 때 최적의 메모리 효율성을 나타내는 지점을 도출하기 위한 시뮬레이션 과정은 다음과 같다.

전처리한 데이터셋에서 각 클래스 비율을 기존의 비율과 동일하게 유지하면서 클래스의 샘플 수를 줄여 클래스 간 상대적인 비율을 유지하였다. 이때 전체 데이터셋을 100%으로 두고, 0.1%부터 0.9%까지는 0.1% 간격으로, 1%부터 10%까지는 1% 간격으로, 10%부터 90%까지는 10% 간격으로 샘플 비율을 증가시키면서 성능을 평가하였다. 하지만 다른 기법과 달리 샘플을 선택하는 기준에 따라 샘플에 차이가 있으므로 각 비율마다 1,000번 undersampling을 반복하여 그 평균값을 사용하였다.

시뮬레이션 결과에 따르면 모든 class에서 undersampling 비율을 0.1%으로 했을 때 가장 높은 메모리 효율성을 보였다. TABLE XII는 undersampling 비율별 모든 class의 정확도, 메모리 사용량, latency 결과를 나타낸 표이며 FIGURE 8은 메모리 효율성을 평가한 결과이다.

TABLE XII

Undersampling results by sampling ratio

Performance metric	Accuracy of class 1	Accuracy of class 2	Accuracy of class 3	Memory usage (kb)	Latency (s)
0.1%	1.8	10.35	92.46	34.67	21.66
0.2%	3.21	8.93	93.01	71.19	22.33
0.3%	3.55	7.14	94.92	105.71	22.96
0.4%	7.17	8.57	92.43	142.24	23.42
0.5%	6.98	6.8	94.97	177.73	24.44
0.6%	5.25	6.69	97.96	213.21	25.5
0.7%	4.29	6.95	97.13	249.74	24.98
0.8%	3.91	7.28	98.1	285.23	25.83
0.9%	4.34	7.9	98.03	320.72	26.37
1%	4.52	8.79	97.47	355.2	27.77
2%	6.44	9.14	97.31	713.25	34.81
3%	9.77	9.71	96.38	1071.55	40.78
4%	13.41	11.5	95.51	1430.38	48.28
5%	16.26	13.24	93.52	1788.62	56.03
6%	17.88	16.05	93.63	2148.46	63.11
7%	18.47	17.78	92.95	2507.01	73.43
8%	19.38	20.49	92.19	2867.54	82.96
9%	20.98	23.87	91.18	3225.74	88.96
10%	23.87	27.83	88.95	3586.93	101.52
20%	43.6	49.45	73.72	7188.27	182.25
30%	54.84	65.63	64.77	10789.26	261.84
40%	56.82	68.43	62.28	14392.61	344.4
50%	61.38	72.22	62.36	17994.87	401.97
60%	66.32	74.09	63.52	21598	471
70%	67.86	75.19	66.3	25200.17	527.17
80%	70.09	76.58	68.46	28806.38	587.67
90%	71.19	77.14	70.94	32412.95	636.27

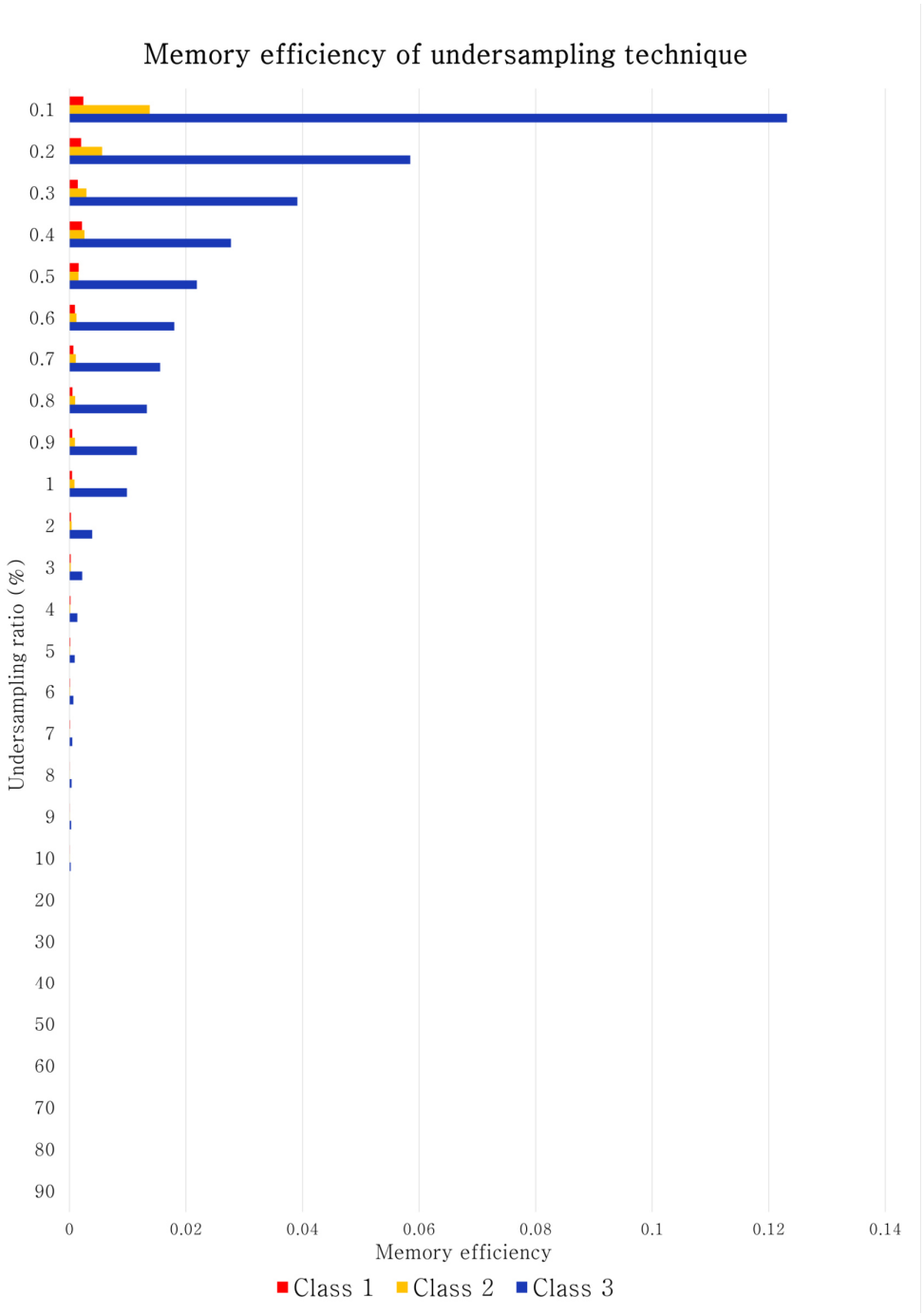

FIGURE 8. Memory efficiency of undersampling technique

데이터 축소 기법 조합의 성능 평가 시뮬레이션은 다음과 같은 과정으로 진행된다. CIC-Malmem-2022 데이터셋에 class 1이 최적의 메모리 효율성을 나타내는 파라미터 값을 활용하여 각각의 데이터 축소 기법을 적용한 새로운 데이터셋을 생성한다. 이때 class 2와 class 3은 고려하지 않는다. Feature selection 기법은 class 1이 중요도가 높은 feature 2개를 사용할 때 메모리 효율이 최적이므로 중요도가 높은 feature 2개를 가진 데이터셋을 생성한다. Bit scaling 기법은 2bit에서 메모리 효율이 최적이므로 기존 데이터셋에 2bit scaling을 적용한 데이터셋, Undersampling 기법은 데이터셋을 0.1%로 undersampling한 데이터셋을 각각 생성한다. 이렇게 생성한 3개의 데이터셋에 다른 데이터 축소 기법을 추가로 적용하여 데이터 축소 기법 간 조합할 수 있는 모든 조합을 생성한다. 마찬가지로 다른 데이터 축소 기법들을 적용할 때 파라미터는 class 1이 최적의 메모리 효율 지점을 나타내는 파라미터를 사용한다.

따라서 하나의 데이터 축소 기법을 적용한 3개의 데이터셋과 2개 이상의 데이터 축소 기법을 적용한 12개의 데이터셋을 생성하여 총 15개의 데이터셋을 생성하였다. 따라서 해당 시뮬레이션에서는 원본 데이터셋과 데이터 축소 기법을 적용한 15개의 데이터셋에 대한 성능을 평가하였다.

데이터 축소 기법과 데이터 축소 기법 조합의 성능 평가 및 분석을 위해 TABLE XIII과 같이 표기법을 정의하였다. Feature selection 기법은 FS로 표현하고, Undersampling 기법은 US, Bit scaling 기법은 BS로 간단하게 표현하며 2개 이상의 데이터 축소 기법을 조합하여 사용한 경우는 “+”로 표기하여 설명한다.

TABLE XIII

Description of the notation

Notation	Description
FS	Feature selection
US	Undersampling
BS	Bit scaling
+	Combination of techniques

시뮬레이션 결과에 따르면 정확도가 가장 높은 기법은 Feature selection 기법이며, 메모리를 가장 적게 사용하는 조합은 3가지 기법을 모두 사용하는 조합 중 FS + US + BS, US + FS + BS, US + BS + FS 조합이었다. latency가 가장 짧은 기법은 Undersampling 기법이었다. TABLE XIV는 CIC-Malmem-2022 데이터셋과 데이터 축소 기법을 적용한 15개의 데이터셋을 사용한 경우의 성능 평가 결과를 나타낸 표이다.

TABLE XIV

Performance evaluation results of combinations

Performance metric	Accuracy (%)	Memory usage (kb)	Latency (s)
CIC-Malmem-2022	71.65	15724.03	178.06
FS	**86.09**	629.49	87.68
US	52.86	34.67	**23.43**
BS	67.38	6467.4	80.35
FS+US	16.15	1.4	24.27
FS+BS	85.87	400.61	88.6
US+FS	38.89	0.94	26.84
US+BS	36.98	4.88	24.82
BS+FS	85.87	400.61	100.54
BS+US	0	30.21	24.48
FS+BS+US	54.17	1.43	25.7
FS+US+BS	6.42	**0.27**	23.27
BS+FS+US	54.17	1.43	27.16
BS+US+FS	23.7	0.94	27.64
US+FS+BS	31.34	**0.27**	27.52
US+BS+FS	31.34	**0.27**	34.17

종래 방식과 제안하는 알고리즘의 메모리 효율성 비교 시뮬레이션은 다음과 같은 과정으로 진행된다.

종래 방식 및 제안하는 알고리즘의 메모리 효율성의 평균값을 평가하기 위해 TABLE XIV의 데이터 축소 기법과 정확도, 메모리 사용량, latency 값을 측정하여 비교하였으며 시뮬레이션 과정을 그림으로 나타내면 FIGURE 9와 같다.

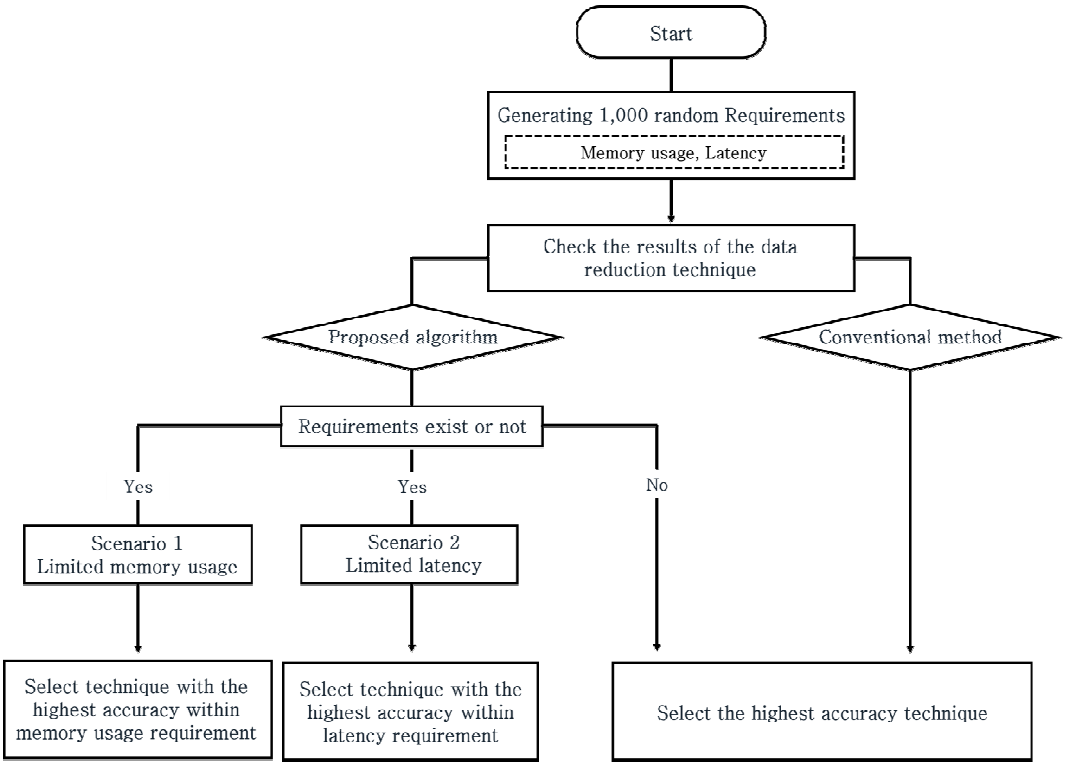

FIGURE 9. Simulation process using CIC-Malmem-2022 dataset

컴퓨팅 환경 및 요구사항을 반영하기 위해 시나리오 1, 시나리오 2를 각각 메모리 사용량과 latency가 제한된 상황으로 정의하고 메모리 사용량, latency의 요구사항을 무작위로 1,000번 생성하였다.

종래 방식은 정확도가 가장 높은 기법을 선택하기 때문에 TABLE XIV의 값 중 가장 높은 정확도를 갖는 데이터 축소 기법을 선택한다. 이와 달리 제안하는 알고리즘은 요구사항을 반영하여 데이터 축소 기법을 동적으로 선택하는 방식이므로 시나리오에 맞는 최적의 기법을 선택한다. 시나리오 1과 같이 메모리 사용량이 제한된 환경에서는 메모리 사용량 요구사항을 만족하는 기법 중 가장 높은 정확도를 갖는 기법을 선택하며, 짧은 latency가 요구되는 시나리오 2의 경우 latency 요구사항을 만족하는 기법 중 가장 높은 정확도를 갖는 기법을 선택한다.

시뮬레이션 결과에 따르면 종래 방식 평균 메모리 효율성은 0.001이었고, 제안하는 알고리즘의 평균 메모리 효율성은 0.09로 메모리 효율성을 약 90배 개선하였다. FIGURE 10은 종래 방식과 제안하는 방식의 메모리 효율성 결과를 비교한 그래프이다.

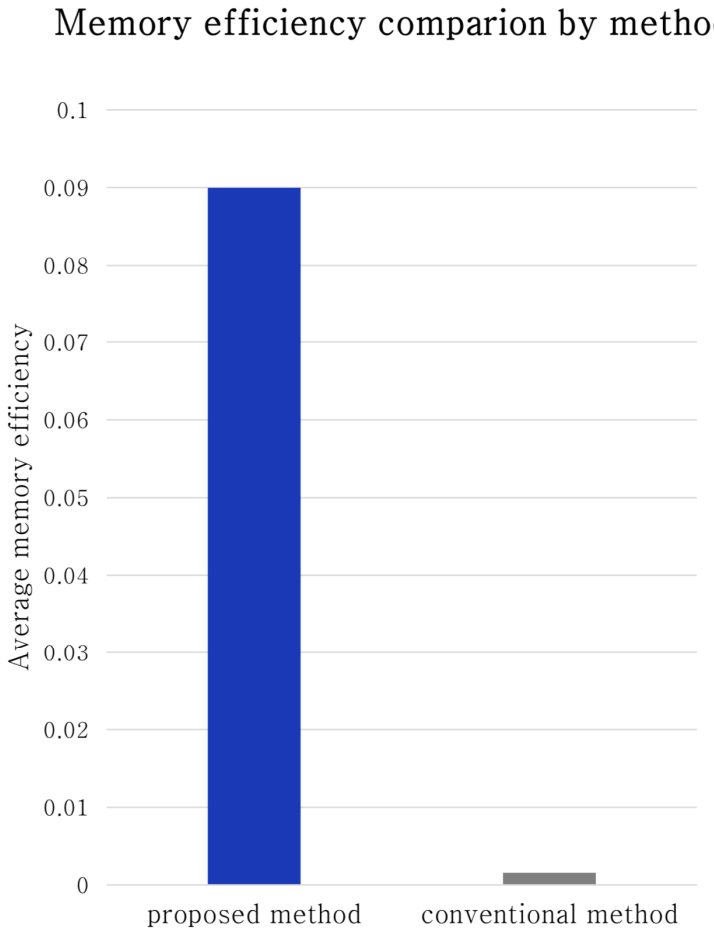

FIGURE 10. Memory efficiency comparison

2) 랜덤 환경 조건 활용 시뮬레이션

두 번째 시뮬레이션은 환경 조건 값으로 정확도, 메모리 사용량, latency 값을 무작위하게 설정하여 제안하는 알고리즘의 효과를 평가했다. 시뮬레이션 과정을 플로우차트로 표현하면 FIGURE 11과 같다.

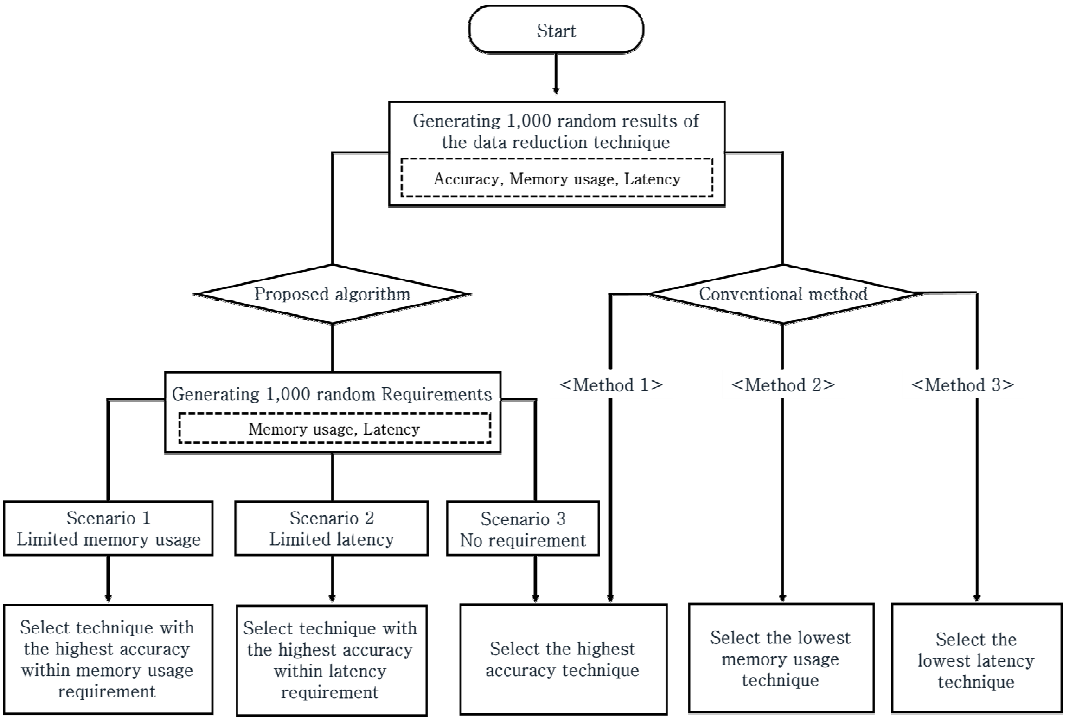

FIGURE 11. Simulation process using random data

먼저 데이터 축소 기법을 적용한 뒤 악성 행위 탐지 성능 평가 결과값을 의미하는 (정확도, 메모리 사용량, latency) 데이터 쌍을 1,000번 무작위하게 생성하였으며, 이 데이터 쌍은 0~1 사이의 값으로 생성된다. 이때 생성된 1,000개의 데이터 쌍은 세 가지의 시나리오로 분류되며 시나리오 1은 메모리 사용량이 제한된 상황, 시나리오 2는 짧은 latency가 필요한 상황을 의미하며 시나리오 3은 요구사항이 없는 상황을 의미한다.

시뮬레이션은 동일한 환경 조건에서 세 개의 Method를 적용했을 때와 제안하는 알고리즘의 메모리 효율성을 평가하고 비교하였다. Method 1은 정확도, 메모리 사용량, latency의 세 가지 평가지표 중에서 정확도가 가장 높은 기법을 선택하고, Method 2는 메모리 사용량이 가장 적은 기법을 선택하며 Method 3은 latency가 가장 짧은 기법을 선택한다. 반면 제안하는 알고리즘은 시나리오에 맞게 제한된 메모리 사용량 또는 latency를 고려하여 동적으로 Method를 선택한다. 요구사항 중 메모리 사용량 요구사항이 있다면 메모리 사용량 요구사항을 만족하는 기법 중 가장 높은 정확도를 갖는 기법을 선택하며 latency 요구사항이 있는 경우에는 이를 만족하는 기법 중 가장 높은 정확도를 갖는 기법을 선택한다. 요구사항이 없는 경우에는 Method 1과 동일하게 가장 높은 정확도를 갖는 기법을 선택한다. 이 시뮬레이션 과정을 5번 반복한 뒤 제안하는 알고리즘과 각 Method별 메모리 효율성을 계산하여 비교한다. 시뮬레이션 결과에 따르면 종래 방식의 경우 평균 8.8의 메모리 효율성 값을 보였지만, 제안하는 알고리즘은 평균 67.6으로 메모리 효율성이 약 7.6배 향상되었다. 즉 동적인 데이터 축소 방식 선택 알고리즘은 종래 방식인 Method 1과 Method 2, Method 3 대비 메모리 효율성을 효과적으로 개선하였다. 해당 시뮬레이션 결과는 TABLE XV와 같다. 또한 FIGURE 12는 메모리 효율성을 비교한 그래프이며, 그래프에서 각 도형은 5번 시뮬레이션을 반복했을 때 메모리 효율성 값을 의미한다.

TABLE XV

Memory efficiency values by method

Iteration	Method 1	Method 2	Method 3	Proposed algorithm
Iteration 1	8.24	42.26	8.24	152.37
Iteration 2	1.27	4.433	7.84	69.09
Iteration 3	19.20	4.29	19.20	40.50
Iteration 4	1.54	0.97	1.31	40.42
Iteration 5	9.78	0.57	3.94	35.94

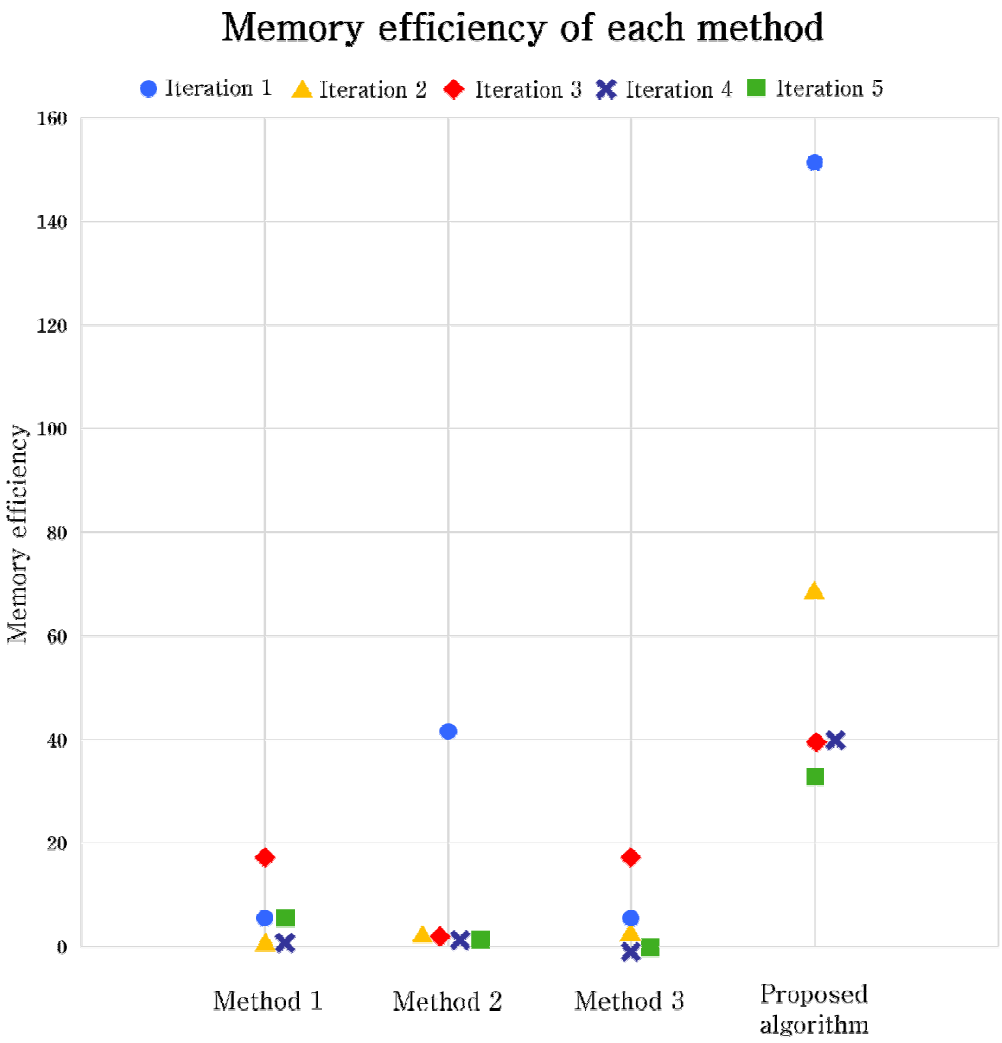

FIGURE 12. Memory efficiency comparison

V. 결론 및 향후 연구

사이버 공격이 점차 지능화되고 다양해지면서 이를 탐지하고 대응하기 위한 인공지능 기반 악성 행위 탐지 기법에 대한 중요성이 커지고 있다. 그러나 일반적인 악성 행위 탐지 기법은 실시간으로 악성 행위를 탐지하기 위해 많은 데이터와 컴퓨팅 자원을 요구하므로 메모리 사용량이나 latency 등 자원이 제약된 환경에 적용하기에 부적합하다. 따라서 자원 제약적인 환경에서도 활용할 수 있는 경량 악성 행위 탐지에 대한 필요성이 대두되고 있다.

본 연구에서는 이러한 문제점을 해결하기 위해 세 가지 데이터 축소 기법을 활용하여 동적인 악성 행위 경량 탐지 데이터 축소 기법 알고리즘을 제안하였다. 제안한 방법은 메모리 효율적인 악성 행위 탐지를 위해 컴퓨팅 환경 및 요구사항을 파악하고, 요구사항이 없는 경우에 가장 높은 정확도를 갖는 기법을 선택하고 메모리 사용량이 제한된 환경 또는 짧은 latency를 요구하는 환경일 경우 해당 요구사항을 만족하는 데이터 축소 기법 중 가장 높은 정확도를 갖는 기법을 선택한다.

시뮬레이션 결과에 따르면 종래 방식은 약 0.001의 정확도 대비 메모리 효율성 값을 보였으나 제안하는 알고리즘의 메모리 효율성 값은 약 0.09로, 메모리 효율성을 약 90배 개선하였다. 또한 다양한 환경 요구사항으로 진행한 메모리 효율성 비교 시뮬레이션에서도 종래 방식의 메모리 효율성 대비 제안하는 알고리즘의 메모리 효율성은 약 7.6배의 향상된 결과를 보였다.

결과적으로 제안하는 알고리즘은 동적으로 데이터 축소 기법을 선택하므로 종래 방식 대비 메모리 효율성을 개선하였으며 메모리 사용량 및 latency가 제약적인 환경에서도 악성 행위를 효율적으로 탐지할 수 있었다. 또한 제한된 컴퓨팅 환경을 만족하는 데이터 축소 기법 중 가장 높은 정확도를 갖는 기법을 선택하여 정확도, 메모리 사용량, latency 간의 트레이드오프 문제를

해결하였다. 향후에는 다양한 형식의 데이터셋을 대상으로 시뮬레이션을 진행하여 제안하는 알고리즘에 대해 신뢰성 있는 결과를 도출할 계획이다.

참 고 문 헌

[1] C. Dan and A. Rezaeipanah, "A systematic survey of data mining and big data analysis in internet of things," The Journal of Supercomputing, vol. 78, no. 17, pp. 18405-18453, Nov. 2022. DOI: 10.1007/s11227-022-04594-1

[2] Statista, "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025," Statista, 2022. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

[3] Quick Heal, "Annual Threat Report 2019," [Online]. Available: https://www.quickheal.co.in/documents/threat-report/QH-Annual-Threat-Report-2019.pdf.

[4] U.-H. Tayyab, F. B. Khan, M. H. Durad, A. Khan, and Y. S. Lee, "A Survey of the Recent Trends in Deep Learning Based Malware Detection," Journal of Cybersecurity and Privacy, vol. 2, no. 4, pp. 800?829, Sep. 2022, doi: 10.3390/jcp2040041.

[5] T. F. Blauth, O. J. Gstrein and A. Zwitter, "Artificial Intelligence Crime: An Overview of Malicious Use and Abuse of AI," in IEEE Access, vol. 10, pp. 77110-77122, 2022, doi: 10.1109/ACCESS.2022.3191790.

[6] T. Hasanin, T. M. Khoshgoftaar, J. Leevy and N. Seliya, "Investigating Random Undersampling and Feature Selection on Bioinformatics Big Data," 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), Newark, CA, USA, 2019, pp. 346-356, doi: 10.1109/BigDataService.2019.00063.

[7] S. Shukla, P. D. Sai Manoj, G. Kolhe and S. Rafatirad, "On-device Malware Detection using Performance-Aware and Robust Collaborative Learning," 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2021, pp. 967-972, doi: 10.1109/DAC18074.2021.9586330.

[8] E. S. Alomari et al., "Malware Detection Using Deep Learning and Correlation-Based Feature Selection," Symmetry, vol. 15, no. 1, p. 123, Jan. 2023, doi: 10.3390/sym15010123.

[9] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, "Android malware classification using optimum feature selection and ensemble machine learning," Internet of Things and Cyber-Physical Systems, vol. 3, pp. 100-111, 2023, ISSN 2667-3452, https://doi.org/10.1016/j.iotcps.2023.03.001.

[10] Y. Yang, L. Deng, S. Wu, T. Yan, Y. Xie, and G. Li, "Training high-performance and large-scale deep neural networks with full 8-bit integers," Neural Networks, vol. 125, pp. 70-82, 2020, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2019.12.027.

[11] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training Deep Neural Networks with 8-Bit Floating Point Numbers," in Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 7686&7695.

[12] B. S. Sharmila and Rohini Nagapadma, "Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset," Cybersecurity, vol. 6, no. 1, pp.

41, 2023, ISSN 2523-3246, https://doi.org/10.1186/s42400-023-00178-5.

[13] L. Cambier, A. Bhiwandiwalla, T. Gong, M. Nekuii, O. Elibol, and H. Tang, "Shifted and Squeezed 8-bit Floating Point format for Low-Precision Training of Deep Neural Networks," 2020.

[14] H. Studiawan and F. Sohel, "Performance Evaluation of Anomaly Detection in Imbalanced System Log Data," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 2020, pp. 239-246, doi: 10.1109/WorldS450073.2020.9210329.

[15] M. Saripuddin, A. Suliman, S. S. Sameon, and B. N. Jorgensen, "Random Undersampling on Imbalance Time Series Data for Anomaly Detection," in *Proceedings of the 2021 4th International Conference on Machine Learning and Machine Intelligence* (MLMI '21), New York, NY, USA, 2022, pp. 151&156, doi: 10.1145/3490725.3490748.

[16] N. Abedzadeh and M. Jacobs, "A Reinforcement Learning Framework with Oversampling and Undersampling Algorithms for Intrusion Detection System," Applied Sciences, vol. 13, no. 20, p. 11275, Oct. 2023, doi: 10.3390/app132011275.

[17] QA4AI Consortium, "Guidelines for Quality Assurance of AI-based Products and Services," QA4AI, Sep. 2021.

[18] B. W. Yap, N. S. M. Ibrahim, H. A. Hamid, S. A. Rahman, and S. J. Fong, "Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy," Pertanika J. Sci. & Technol., vol. 26, pp. 329 - 340, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:67348356

[19] M. S. S. Sumi and A. Narayanan, "Improving Classification Accuracy

Using Combined Filter+Wrapper Feature Selection Technique," in Proc. 2019 IEEE Int. Conf. on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-6, doi: 10.1109/ICECCT.2019.8869518.

[20] A. Kaur, K. Guleria, and N. K. Trivedi, "Feature Selection in Machine Learning: Methods and Comparison," in Proc. 2021 Int. Conf. on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2021, pp. 789-795, doi: 10.1109/ICACITE51222.2021.9404623.

[21] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar and N. Seliya, "IoT Reconnaissance Attack Classification with Random Undersampling and Ensemble Feature Selection," 2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC), Atlanta, GA, USA, 2021, pp. 41-49, doi: 10.1109/CIC52973.2021.00016.

[22] M. Zeng, B. Zou, F. Wei, X. Liu and L. Wang, "Effective prediction of three common diseases by combining SMOTE with Tomek links technique for imbalanced medical data," 2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS), Chongqing, China, 2016, pp. 225-228, doi: 10.1109/ICOACS.2016.7563084.

[23] N. M. Mqadi, N. Naicker, T. Adeliyi, and J. Hemanth, "Solving Misclassification of the Credit Card Imbalance Problem Using Near Miss," Math. Probl. Eng., vol. 2021, Art. no. 7194728, Jul. 20, 2021. doi: 10.1155/2021/7194728. [Online]. Available: https://doi.org/10.1155/2021/7194728

[24] N. Verdikha, T. Adji, and A. Permanasari, "Study of Undersampling Method: Instance Hardness Threshold with Various Estimators for Hate S

peech Classification," IJITEE (Int. J. Inf. Technol. Electr. Eng.), vol. 2, Dec. 26, 2018. doi: 10.22146/ijitee.42152.

[25] S. Buschjager and K. Morik, "Improving the Accuracy-Memory Trade -Off of Random Forests Via Leaf-Refinement," 2021. [Online]. Availabl e: https://arxiv.org/abs/2110.10075. arXiv: 2110.10075

[26] R. Zuech, J. Hancock, and T. M. Khoshgoftaar, "Detecting web attacks using random undersampling and ensemble learners," *Journal of Big D ata*, vol. 8, no. 1, p. 75, May 2021, doi: 10.1186/s40537-021-00460-8.

[27] UNB CIC,"CIC Malware 2022 Dataset," University of New Brunswick. [Online].Available: https://www.unb.ca/cic/datasets/malmem-2022.html.

ABSTRACT

Dynamic Data Reduction Technique Selection Algorithm for Lightweight Detection of Malicious Behavior

Min-Jeong Kim
Department of Future Convergence Technology Engineering
Graduate School of
Sungshin Women's University

Recently advancements in malware have seen a diversification in attack methodologies and an increase in sophistication, prompting active research into AI-based malware detection techniques. Accurate detection of malicious activities necessitates large volumes of data for AI training and inference. However, there is a problem that storing and learning big data collected in real time requires high-performance hardware and substantial memory capacity. In particular, in an environment where available resources are limited, such as IoT, it is difficult to apply general malicious behavior detection techniques, resulting in trade-off problems in accuracy, memory usage, and latency.This study proposes an optimal data reduction

technique by optimizing feature selection, bit scaling, and undersampling technique to solve the issue of memory inefficiency in malicious behavior detection within big data environments. An algorithm is proposed that dynamically selects a data reduction technique suitable for the computing environment, effectively improving memory efficiency relative to accuracy. Experimental results show that the proposed method offers approximately 90 times higher memory efficiency compared to conventional methods, and an improvement of about 7.6 times in memory efficiency is demonstrated under randomly generated environmental conditions. Therefore, the proposed algorithm showed that it is possible to detect malicious behavior by reducing memory usage and latency generated in the process of storing and learning existing vast amounts of data.