Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.5, Oct. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.5.761

활성화 함수 근사를 통한 지수함수 기반 신경망 마스킹 기법*

김 준 섭,[1†] 김 규 상,[1] 박 동 준,[1] 박 수 진,[1] 김 희 석,[2] 홍 석 희[2‡]
[1,2]고려대학교 (대학원생, 교수)

Masking Exponential-Based Neural Network via Approximated Activation Function*

Joonsup Kim,[1†] GyuSang Kim,[1] Dongjun Park,[1]
Sujin Park,[1] HeeSeok Kim,[2] Seokhie Hong[2‡]
[1,2]Korea University (Graduate student, Professor)

요 약

본 논문에서는 딥러닝 분야에서 사용되는 신경망 모델, 그중에서도 다중 계층 퍼셉트론 모델에 사용되는 지수함수 기반의 활성화 함수를 근사 함수로 대체하고, 근사 함수에 마스킹을 적용함으로써 신경망 모델의 추론 과정의 전력 분석 저항성을 높이는 방법을 제안한다. 이미 학습된 값을 사용하여 연산하는 인공 신경망의 추론 과정은 그 특성상 가중치나 편향 등의 내부 정보가 부채널 공격에 노출될 위험성이 있다. 다만 신경망 모델의 활성화 함수 계층에서는 매우 다양한 함수를 사용하고, 특히 지수함수 기반의 활성화 함수에는 마스킹 기법 등 통상적인 부채널 대응기법을 적용하기가 어렵다. 따라서 본 연구에서는 지수함수 기반의 활성화 함수를 단순한 형태로 근사하여도 모델의 치명적인 성능 저하가 일어나지 않음을 보이고, 근사 함수에 마스킹을 적용함으로써 전력 분석으로부터 안전한 순방향 신경망 모델을 제안하고자 한다.

ABSTRACT

This paper proposes a method to increase the power-analysis resistance of the neural network model's feedforward process by replacing the exponential-based activation function, used in the deep-learning field, with an approximated function especially at the multi-layer perceptron model. Due to its nature, the feedforward process of neural networks calculates secret weight and bias, which already trained, so it has risk of exposure of internal information by side-channel attacks. However, various functions are used as the activation function in neural network, so it's difficult to apply conventional side-channel countermeasure techniques, such as masking, to activation function(especially, to exponential-based activation functions). Therefore, this paper shows that even if an exponential-based activation function is replaced with approximated function of simple form, there is no fatal performance degradation of the model, and than suggests a power-analysis resistant feedforward neural network with exponential-based activation function, by masking approximated function and whole network.

Keywords: AI, Activation Function, Deep Learning, Side Channel Analysis, Masking

Received(09. 07. 2023), Modified(10. 10. 2023), Accepted(10. 10. 2023)

* 본 논문은 2023년도 정보보호학회 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임.

* 본 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00903, 고신뢰 온-디바이스 딥러닝 가속기 설계를 위한 물리채널 기반 취약점 검증 및 대응기술 개발)

† 주저자, kimjs0768@korea.ac.kr

‡ 교신저자, shhong@korea.ac.kr(Corresponding author)

I. 서 론

최근 무인 자동차나 ChatGPT 등으로 대표되는 인공지능과 머신러닝 분야에 관한 관심이 높아지면서 딥러닝에 관한 연구가 활발하게 진행되고 있다. 딥러닝이란 이미지의 분류작업과 같이 인간이 쉽게 처리할 수 있는 작업을 기계로 재현하기 위해 사용되는 인공 신경망 모델이 발전된 형태로, 여러 계층을 가중치로 연결하고 오류의 역전파(back-propagation)를 통해 가중치를 조정함으로써 복잡한 데이터를 학습할 수 있도록 하는 기술이다[1].

오늘날 딥러닝 기술이 광범위하게 활용되면서 다양한 기기에 신경망 모델이 구현되는 경우가 증가하고, 이에 따라 공격자의 기기 접근이 쉬워지면서 딥러닝 기술에 사용되는 신경망 모델의 부채널 공격 취약성 문제가 대두되었다. 신경망 모델은 기본적으로 이러한 공격에 대응할 수 있도록 설계되지 않았기 때문에, 부채널 분석을 통해 모델의 정보를 추출하거나 모델의 구조 또는 파라미터가 복원될 가능성이 있다[2, 3]. 딥러닝 기술이 활용되는 자율주행, 의료시스템 등 인간의 생명과 직결된 분야에서의 안전 문제를 고려했을 때, 신경망 모델의 부채널 취약점은 결코 무시할 수 없는 위협이며, 딥러닝 신경망 모델에 대한 부채널 공격 대응기법 연구의 중요성이 커지고 있다.

이에 본 논문에서는 기존에 사용하던 지수함수 기반의 활성화 함수를 다항식 기반 근사 함수로 대체하고, 부채널 공격에 대응하기 위해 마스킹 기법을 적용하는 알고리즘을 제시할 것이다. 또한 이러한 알고리즘이 실제로 전력 분석 공격에 안전한 추론 신경망을 구성할 수 있음을 실험을 통해 검증할 것이다. 이때, 학습된 가중치가 부채널 공격에 노출되지 않도록 보호하려면 끊임없이 가중치가 변화하는 학습 과정보다는, 학습이 완료된 가중치를 이용하며 다양한 기기에 탑재되어 공격자가 접근, 또는 습득하여 부채널 정보를 얻기 쉬운 순방향(feedforward) 과정에 대응기법을 적용할 필요가 있다. 따라서 본 연구는 마스킹의 적용 범위를 신경망의 추론 과정(순방향 연산)으로 설정하고 진행되었다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 부채널 분석과 마스킹 기법, MLP 모델과 활성화 함수에 대한 기본적인 개념을 소개하고, 3장에서 시그모이드(sigmoid)와 쌍곡탄젠트(tanh), 두 종류의 활성화 함수를 다양한 근사법과 다항식으로 근사한 함수를 제시하고, 해당 함수를 추가적으로 조정하여 활성화 함수를 대체할 수 있을지를 검증한다. 이후 4장에서는 MNIST 데이터 셋을 활용하는 MLP 모델을 구성하여 시그모이드 함수를 사용해 학습을 진행하고, 학습된 가중치를 사용해 추론 테스트를 진행했을 경우 원본 함수와 근사 함수의 정확도를 비교하여, 활성화 함수를 근사 함수로 대체할 수 있음을 제시한다. 5장에서는 해당 근사 함수를 사용하는 신경망 구성에 필요한 연산에 마스킹을 적용하는 알고리즘을 제시하며, 6장에서 마스킹 알고리즘의 안전성 실험 및 마스킹된 순방향 신경망의 성능 실험을 진행한 결과를 확인한다. 마지막으로 7장에서는 실험 결과에서 얻을 수 있는 결론을 정리하고, 향후 연구 방향을 소개하며 마친다.

II. 배경 지식

2.1 부채널 분석과 마스킹 기법

모든 전자 기기는 데이터 연산 중 컴퓨팅 장치의 논리 회로의 활동에 따라 연산 시간, 전력 소비량, 전자기파 등 의도되지 않은 부차 신호를 생성한다. 이러한 부차 신호는 회로에서 처리하는 데이터의 중간값들에 의존하기 때문에, 이를 통계적으로 분석하여 중간값을 추론하고, 수학적으로 안전하다고 알려진 알고리즘으로부터 비밀값을 추출할 수 있다. 이러한 접근 방식을 부채널 분석(Side-Channel Analysis)이라 하며, 부채널 분석은 암호학뿐만 아니라 인공지능 분야에서 또한 그 중요성이 커지고 있다[3].

가장 일반적인 부채널 공격 대응기법 중 하나인 마스킹 기법은 알고리즘에서 연산에 사용하는 값에 난수를 XOR하거나 덧셈함으로써 중간값이 그대로 노출되는 것을 방지하고, 부채널 공격을 통한 비밀값 추출을 무력화하는 기법이다. 가장 일반적인 마스킹 기법으로는 산술(arithmatic) 마스킹과 부울(boolean) 마스킹이 있는데, 산술 마스킹은 환 위에서의 덧셈 연산을 이용해 난수를 더하는 방식이고 부울 마스킹은 배타적 논리합(XOR)을 이용해 난수를 더하는 방식이다. 이를 응용하여 하나의 변수를 여러 개로 나누어 저장하는 (t+1)-sharing 방식도 존재한다. 유한환 $\mathbb{Z}_L$의 원소가 되는 난수 $x^0, x^1, \ldots x^{t-1}$을 이용하여 변수 x를

$< x > = < x^0, x^1, \ldots, x^t >$ 형태의 튜플로 저장하는 방식으로, 이 때 해당 튜플의 원소들은 $x^0 \bigstar x^1 \bigstar \ldots \bigstar x^t = x$를 반드시 만족해야 한다. ($\bigstar$는 환 위의 덧셈 또는 XOR 연산이다.)

신경망 모델에서 또한 활성화 함수의 입출력에서 본래의 입력값이 노출되지 않도록 마스킹 기법을 적용하는 부채널 공격 대응 방식을 고려할 수 있는데, 신경망 모델에서는 실수를 변수로 사용하는 경우가 많아 기존 암호학 분야에서의 마스킹 기법을 그대로 적용하기가 어려우며 활성화 함수의 다양성 탓에 일괄적으로 마스킹을 적용하기가 어렵다는 문제가 있다. 실수 변수의 경우 고정 소수점 방식을 이용하여 실수 연산을 정수 연산으로 변환한 후, ReLU(Recified Linear Unit) 활성화 함수를 사용하는 마스킹된 신경망을 제시한 선행 연구가 있지만[5], 지수함수 기반의 활성화 함수에 해당 방식을 적용할 수 없다는 점 등 한계가 존재한다.

2.2 고정 소수점 연산

고정 소수점 방식이란 실수를 비트로 나타내는 방법의 하나로, 소수부를 표현하기 위해 고정된 숫자의 비트를 할당하여 사용한다. 부동 소수점 방식에 비해 표현할 수 있는 범위는 좁지만, 실수를 정수 형태로 나타낼 수 있으며 모든 연산을 정수 형태로 치환할 수 있다는 장점이 있다. 예를 들어 고정소수점의 덧셈은 단순한 정수의 덧셈으로 치환할 수 있으며, 고정소수점의 곱셈은 정수 간의 곱셈을 수행한 뒤 소수점을 맞추기 위해 하위 비트를 버리는 시프트 연산을 수행해 나타낼 수 있다.

부동 소수점 방식은 사용하는 경우 표현할 수 있는 범위는 넓지만, 연산의 정밀도가 떨어지고 정상적으로 난수를 적용하기 어렵기 때문에 마스킹을 적용하기 어렵다. 반면 고정 소수점 방식은 실수 연산을 정수 연산으로 치환할 수 있으므로, 기존의 마스킹 방식을 쉽게 적용할 수 있으며 연산이 단순해진다. 따라서 실수 변수에 마스킹을 적용할 때 일반적으로 고정 소수점 방식을 이용한다.

2.3 MLP 모델과 활성화 함수

MLP 모델은 신경망 모델의 한 종류로, 그림 1과 같이 입력 계층과 은닉 계층, 출력 계층을 포함한 3개 이상의 계층(layer)과 각 계층을 구성하는 인공

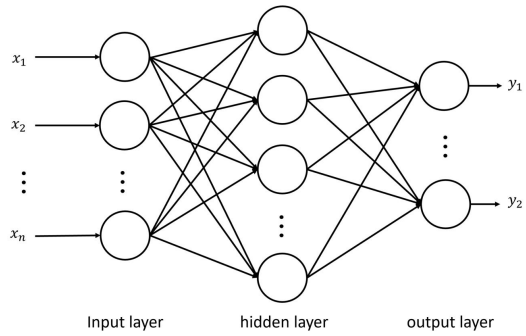


Fig. 1. Basic Structure of MLP Network.

뉴런에 해당하는 노드들로 이루어져 있다. 각각의 노드는 인접한 계층의 모든 노드와 연결되어 있으며, 각 노드에는 이전 계층의 출력값에 각각의 가중치를 곱한 값의 합과 편향을 더한 값이 입력되어, 입력값에 활성화 함수를 적용한 값이 출력된다.

$$y = Activation(\sum(weight * x) + bias) \qquad (1)$$

여기서 활성화 함수는 신경망 모델에서 입력 신호의 총합을 출력 신호로 변환하는 함수로, 인간의 뉴런에서 자극이 임계점(threshold)을 넘어야만 반응이 일어나는 것과 마찬가지로, 특정 값을 넘어서는 입력 신호에 대해서만 출력 신호를 생성하도록 하는 역할을 한다.

활성화 함수로 선형 함수를 사용하면 신경망 전체의 입출력이 선형 관계가 되어 단일 계층의 선형 함수와 같아지므로, 은닉 계층을 쌓는 의미가 사라진다. 따라서 활성화 함수에는 비선형 함수를 사용해야 한다. 주로 사용되는 활성화 함수로는 시그모이드 함수 쌍곡탄젠트 함수, ReLU 함수 등이 있으며, 이를 변형한 다양한 활성화 함수가 활용된다. 시그모이드 함수와 쌍곡탄젠트 함수는 입력값을 각각 0과 1 사이, -1과 1 사이의 출력값으로 변환하며, ReLU 함수는 0 이하의 입력값은 0으로, 양의 입력값은 그대로 출력하는 특성이 있다.

III. 활성화 함수의 근사

3.1 활성화 함수의 다항식 근사

함수의 근사는 복잡한 함수를 더 빠르게 계산할 수 있는 다른 함수로 대체하는 기법이다. 자연스럽게

근사 함수와 기존 함수 사이에는 어느 정도의 오차가 존재하므로, 모델의 분류 오차는 다소 증가할 수 있으나, 약간의 오차를 허용할 수 있는 상황에서 복잡한 함수를 근사 함수로 대체함으로써 더욱 빠른 연산을 가능하게 한다. 이러한 함수 근사의 특징을 이용해 복잡한 지수함수 기반의 활성화 함수를 단순한 형태로 변환하면 부채널 대응기법을 설계할 더 쉽게 설계할 수 있을 것이다. 본 연구에서는 활성화 함수의 근사를 위해 널리 알려진 테일러 전개(Taylor Series)와 체비쇼프 전개(Chebyshov Series)를 이용해 시그모이드 함수와 쌍곡탄젠트 함수의 근사 다항함수를 구했다. 그림 2는 근사 함수와 원본 함수의 형태 및 오차를 그래프로 나타낸 것이다.

그림 2에서 볼 수 있듯이, 오차 정도는 원본 함수의 종류와 근사 방법에 따라 차이가 있지만 활성화 함수를 일정 범위 내에서 충분히 작은 오차를 갖는 근사 함수로 근사할 수 있다. 다만 다항함수 자체는 일정 범위 이상에서 급격하게 발산하므로 이를 활성화 함수로 사용하면 손실 값이 지나치게 커져, 은닉 계층의 수가 늘어나는 경우 학습에 이용하기 어렵다. 또한 수렴하는 기존 활성화 함수로 학습한 가중치를 그대로 사용하는 경우 정확한 추론 결과를 얻어내기도 어렵다는 문제 또한 발생한다. 따라서 근사 함수로 활성화 함수를 대체하려면 모델에 추가적인 정규화 과정을 추가하여 오차를 최소화할 수 있는 범위를 벗어나는 값이 활성화 함수에 입력되지 않도록 조절하거나, 다항함수의 입력 또는 출력값을 조정할 필요가 있다.

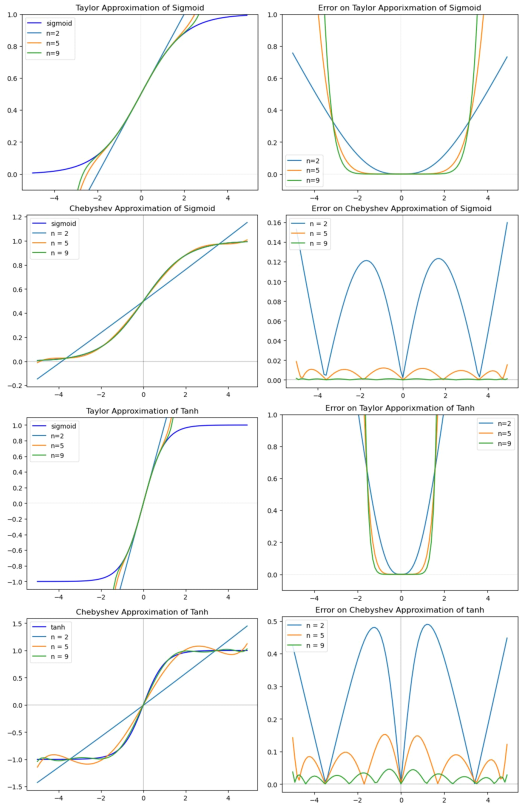


Fig. 2. Graphs of Approximated Activation Function and Error.

3.2 근사 함수의 조정

근사 함수를 추가로 조정하는 경우, 입력을 조정하는 방법과 출력을 조정하는 방법이 있다. 출력을 조정하는 경우, 원본 함수가 수렴하는 값인 ± 1 범위를 벗어나는 값을 $\pm(1-\epsilon)$으로 변경한다. 입력을 조정하는 경우 기존 활성화 함수가 수렴하기 시작하는 부분의 입력값을 임계점으로 설정해두고, 범위를 벗어나는 값이 입력되면 임계점에 함수값을 대신 출력한다. 예를 들어, 활성화 함수 $f(x)$에 대해 입력 임계값을 ± 5로 설정했다면 $-5 \leq x < 5$ 인 x에 대해서는 그대로 $f(x)$를 출력하고, $x \geq 5$ 또는 $x < -5$인 x에 대해서는 각각 $f(5)$와 $f(-5)$를 출력하는 식이다.

이처럼 함수를 조정하면 모든 범위에서 원본 함수와 유사한 양상을 보이는 근사 함수를 얻을 수 있으며, 오차 또한 매우 작은 수준으로 유지할 수 있다. 그림 3은 시그모이드 함수와 출력을 조정한 체비쇼프 5차 근사 함수, 입력을 ± 4.75를 임계점으로 하여 조정한 체비쇼프 5차 근사 함수의 비교 그래프와 오차이다.

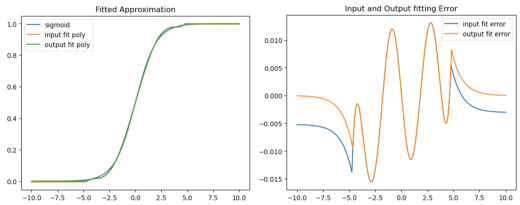


Fig. 3. Graph of Fitted Chebyshev 5'th Approximation of Sigmoid and Error.

IV. MLP 모델 MNIST 데이터 추론 실험

3장에서 확인한 근사 함수를 실제로 활성화 함수 대신 사용할 수 있음을 확인하기 위해, 근사 함수를 사용한 추론 정확도 실험을 진행하였다.

4.1 실험 명세

학습 및 추론 과정에는 MNIST 데이터셋을 사용했다. MNIST는 그림 4와 같이 손으로 쓴 28×28 픽셀 아라비아 숫자의 이미지 데이터로, 딥러닝 분야에서 모델의 학습 및 테스트에 널리 사용되는 데이터이다.

모델의 구성은 표 1과 같으며, 학습은 모두 시그모이드 함수를 사용하여 진행하였다. 첫 번째로 은닉 계층이 한 계층인 경우에 대해 학습된 가중치를 이용하여 조정 방식에 따른 추론 정확도를 측정하여 비교하고, 두 번째로 근사 방법과 차수에 따른 정확도를

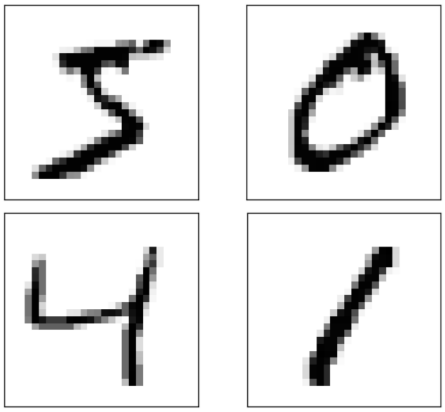


Fig. 4. Mnist Data Images.

Table 1. Model Structure and Parameters.

Used Data	MNIST Dataset
Num of Data	60,000(train), 10,000(test)
Optimizer	Adam
Loss Function	sparse_categorical_crossentropy
Epoch	30
Batch Size	32
Input Nodes	784
Hidden Nodes	128
Output Nodes	10
Metric	Accuracy

Table 2. Environment of Experiment.

Hardware/Software Specification	
CPU	Intel Core i7-13700K @ 3.40GHz
RAM	Samsung DDR4-3200(64GB)
OS	Windows 11 x64
Lang.	Python 3.10.9
Library	TensorFlow 2.12.0

은닉 계층이 한 계층인 경우와 세 계층인 경우에 대해 각각 측정하여 비교하였다.

4.2 추가 조정 방식에 따른 추론 정확도 비교

4.1에서 제시한 모델과 데이터를 이용하여, 각각의 추가 조정 방식에 따른 추론 정확도 차이를 비교하기 위해 체비쇼프 5차 근사 함수를 기준으로 입력을 조정한 근사 함수, 출력을 조정한 근사 함수, 정규화 과정을 추가한 시그모이드 함수와 정규화 과정을 추가한 근사 함수에 대해 각각 학습 주기별 추론 정확도를 측정하여 비교하였다.

그림 5에서 볼 수 있듯이, 아무런 추가 조정을 진행하지 않은 다항식 근사 함수(polynomial)는 매우 낮은 정확도를 보였으며, 입력 또는 출력을 조정한 근사 함수와 원본 시그모이드 함수는 유의미한 차이 없이 높은 정확도를 보였다. 정규화 과정을 추가로 진행하였을 때는 정확도가 감소하여, 모델의 성능이 다소 떨어짐을 확인할 수 있었다.

정규화 과정을 추가하려면 기존에 학습시킨 가중치 대신 정규화 과정이 추가된 새로운 가중치를 학습

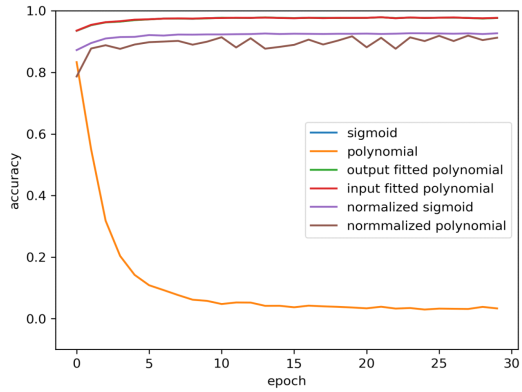


Fig. 5. Accuracy of feedforward inference network using original and approximated activations.

하는 과정이 추가로 필요하므로, 기존의 학습 데이터를 그대로 사용하면서도 높은 정확도를 보이는 근사 함수를 사용하여 추론 신경망을 구성하는 편이 효율적이라 판단할 수 있다.

4.3 근사 차수에 따른 추론 정확도 비교

테일러 전개와 체비쇼프 전개를 이용해 시그모이드 함수를 1차, 5차, 9차 식으로 근사한 다항함수에서 입력을 조정한 근사 함수와 시그모이드 함수로 학습시킨 가중치를 이용해 순방향 연산을 수행한 뒤, 각각의 정확도를 비교하였다.

실험 결과, 표 3에서 볼 수 있듯이, 근사 차수나 방식과 관계없이 기존 함수와 거의 같은 높은 정확도를 보였으며 은닉 계층 수가 늘어나도 여전히 유의미한 정확도 감소는 보이지 않았다. 따라서 신경망 모델의 추론 과정에서 기존의 활성화 함수를 문제없이 대체할 수 있음을 알 수 있고, 이를 이용해 근사 함수에 마스킹을 적용하여 부채널 공격에 안전한 신경망 모델을 설계할 수 있을 것이다.

근사 차수가 높으면 정밀한 계산이 가능하지만, 연산 복잡도가 증가하며 특히 마스킹 적용을 위해 실수를 고정 소수점 방식으로 표현하여 계산하는 경우 고차 다항식 계수가 되는 매우 작은 실수를 나타내기 위해 많은 소수부 비트가 요구되며, 거듭제곱의 연산 결과가 매우 커질 수 있으므로 이를 표현하기 위해 더욱 많은 비트를 사용해야 한다. 이처럼 각각의 장단점이 고려하여 모델의 구조나 사용 환경에 따라 근사 함수를 취사선택하여 사용할 수 있다.

또한 고정 소수점 방식을 사용한다고 가정할 경우, 입력을 조정하는 경우와 출력을 조정하는 경우 사이에 추론 정확도 차이는 없지만 입력값이 커질수록 한정된 비트 내에 거듭제곱의 결과를 정상적으로 저장하기 어려워지므로 정확한 계산을 위해서는 출력보다는 입력을 조정하는 편이 바람직하다.

Table 3. Inference Accuracy of sigmoid and approximated functions.

Activation Function	Accuracy (1 hidden layer)	Accuracy (3 hidden layer)
Sigmoid	0.9753	0.9769
taylor 1	0.9746	0.9768
taylor 5	0.9747	0.9769
taylor 9	0.9752	0.9771
cheb 1	0.9752	0.9725
cheb 5	0.9752	0.9775
cheb 9	0.9750	0.9775

V. 마스킹된 신경망 알고리즘

앞서 소개한 근사 함수에 마스킹을 적용하기 위해서는 마스킹된 덧셈이나 곱셈 등 기본적인 연산에 더해 입력을 조정하기 위한 비교 연산 등이 필요하다.

본 장에서는 안전한 신경망 설계를 위한 연산을 마스킹된 2-sharing 튜플 형태로 수행하는 알고리즘을 제시하고, 동작 원리를 간단히 설명한다.

본 장에서 서술하는 알고리즘에서, $<x>$는 x에 마스킹을 적용한 2-sharing 튜플을 말하며, $<x>_0$와 $<x>_1$은 각각 $<x>$의 첫 번째 원소와 두 번째 원소를 나타낸다.

5.1 마스킹된 덧셈, 곱셈, 버림 알고리즘

덧셈과 곱셈은 신경망의 모든 연산을 위한 기본 연산이며, 2-sharing 형태의 입력값 튜플 두 개를 입력으로 받아 덧셈, 곱셈 결과에 해당하는 튜플을 출력한다. 곱셈은 1차원의 벡터 내적과 같으므로, 일반화를 위해 내적 연산 알고리즘을 설계한다. 이때 각 튜플의 원본 값은 복원되지 않으며, 출력의 난수성을 보존하기 위해 새로운 난수를 추가한다.

고정 소수점 변수를 정확하게 곱하기 위해서는 정수 곱셈을 수행한 뒤, 소수부 비트만큼 하위 비트를 버리는 시프트 연산을 수행해야 한다. 마스킹을 적용하는 경우 부호가 있는 정수를 부호가 없는 정수의 튜플로 나타내고 있으므로 단순히 시프트해서는 올바

Alg.1. Masked Add

Input: two 2-sharing tuples $<x>, <y>$
Output: 2-sharing $<z>$ s.t. $z = x + y$

1. $r \leftarrow rand()$
2. $<z>_0 \leftarrow r, <z>_1 \leftarrow -r$
3. $<z>_0 \mathrel{+}= <x>_0$
   $<z>_1 \mathrel{+}= <x>_1$
4. $<z>_0 \mathrel{+}= <y>_0$
   $<z>_1 \mathrel{+}= <y>_1$
5. $return <z>$

Fig. 6. Masked Addition Algorithm.

Alg.2. Masked Dot(Mul)

Input: two 2-sharing vectors
$< X > = (< X_0 > , < X_1 > , \dots < X_{k-1} >$
$< Y > = (< Y_0 > , < Y_1 > , \dots < Y_{k-1} >$
Output: 2-sharing $< Z >$ s.t.
$Z = X \bullet Y$

1. $r \leftarrow rand()$
2. $< Z >_0 \leftarrow r$, $< Z >_1 \leftarrow -r$
3. **for** i from 0 to $k-1$ **do**
4. $< Z >_0 \mathrel{+}= < X_i >_0 * < Y_i >_1 + < X_i >_1 * < Y_i >_0$
   $< Z >_1 \mathrel{+}= < X_i >_0 * < Y_i >_0 + < X_i >_1 * < Y_i >_1$
5. $return < Z >$

Fig. 7. Masked Dot Product Algoritm.

른 결과값을 얻을 수 없으며, 정확한 결과를 얻기 위해 튜플의 형태에 따라 오버플로우 발생 유무를 확인하여 결과를 조정해야 한다. 하지만 [6]에서 제안한 멀티 파티 컴퓨팅에서의 share 버림 알고리즘을 사용하면 높은 확률로 오차가 거의 없는 정확한 결과를 얻을 수 있다.

알고리즘 3은 $x \in [0, 2^{l_t}] \cup [2^l - 2^{l_t}, 2^l)$을 만족하는 x에 대해 $1 - 2^{l_t+1-l}$의 확률로 오차가 1 이하인 결과를 출력한다. 2^l은 튜플 생성을 위해 산술 덧셈을 진행하는 환의 크기이며, l_d는 버림할 비트 수에 해당한다. x에 해당하는 중간값들은 2^{32}에 비해 매우 작으므로, 시행 횟수를 고려해도 연산이 정상적으로 수행되지 않을 확률이 매우 낮으며, 모델의 성능에 영향을 주지 않는다.

Alg.3. Masked Trunc

Input: 2-sharing $< x >$
Output: 2-sharing $< z >$ s.t.
$z \in \left\{ \left\lfloor \frac{x}{2^{l_d}} \right\rfloor , \left\lfloor \frac{x}{2^{l_d}} \right\rfloor + 1, \left\lfloor \frac{x}{2^{l_d}} \right\rfloor - 1 \right\}$

1. $r \leftarrow rand()$
2. $< z >_0 \leftarrow r$, $< z >_1 \leftarrow -r$
3. $< c >_0 \leftarrow < x >_0 \gg l_d$
   $tmp \leftarrow (2^l - < x >_1) \gg l_d$
   $< c >_1 \leftarrow 2^l - tmp$
4. $< z >_0 \mathrel{+}= < c >_0$
   $< z >_1 \mathrel{+}= < c >_1$
5. $return < z >$

Fig. 8. Masked Truncation Algorithm.

5.2 마스킹된 비교 알고리즘

마스킹된 값을 복원하지 않고 비교하기 위해, 선행 연구에서는 산술 마스킹과 부울 마스킹을 전환하여 MSB를 확인하는 방식을 이용했다. 하지만 산술 마스킹과 부울 마스킹의 전환 알고리즘은 다수의 선형 연산으로 이루어져 있으며, 매우 높은 오버헤드가 발생한다는 단점이 있는 것으로 알려져 있다. 이를 보완하기 위해, [7]에서 제안된 마스킹된 계단 함수를 응용하여 튜플로 저장된 두 정수의 값을 복원하지 않고, 쪼개진 값의 비트만을 확인하여 대소를 비교하는 알고리즘을 알고리즘 4과 같이 설계했다.

먼저 마스킹된 덧셈 알고리즘을 이용해, $x-a$ 에 해당하는 튜플 c를 생성한다. 이때 c는 $(x-a+m, -m)$ 의 형태로 구성되며(m은 난수), 튜플 c로부터 얻은 두 값 $x-a+m$과 m을 한 비트씩 비교하여 뺄셈 결과의 MSB를 추측한다.

양쪽의 비트가 같은 부분은 서로 상쇄되어 최종적인 부호에 영향을 주지 않으므로 양쪽의 비트가 서로 다른 부분에만 주목하면 된다. 최상위 비트를 제외한 나머지 비트를 확인하여 두 비트가 다르고, c^m의 비

Alg.4. Masked Cmp

Input: two 2-sharing tuples $< x >$ and $< a >$
Output: 2-sharing $< z >$ s.t.
z is MSB of $x-a$

1. $< c > := masked\ add(< x > , < -a >)$
2. $c^m := < c >_0$, $m := -< c >_1$
   $r \leftarrow rand() \ll 1$
   $< z >_0 = r$
   $< z >_1 = -< z >_0$
3. **for** i from 4 to $l-2$ **do**
   $cmp = c_i^m \oplus m_i$
   $< z >_0 \leftarrow (cmp \wedge (r + (c_i^m \oplus 1))) \vee ((cmp \oplus 1) \wedge < z >_0)$
4. $cmp = c_{l-1}^m \oplus m_{l-1}$
   $result \leftarrow cmp \oplus result$
5. $return\ result$

Fig. 9. Masked Comparison Algorithm.

트가 1이라면 c^m쪽이 크므로 $x-a$가 양수, 반대라면 음수라고 추측할 수 있다. 이처럼 두 비트가 다른 경우 c^m의 비트에 따라 값을 갱신한다. 양쪽의 MSB가 다른 경우 실제 뺄셈 결과의 부호가 하위 비트의 대소관계와 반대가 되므로, 하위 비트에서 확인한 결과를 반전한다.

단, LSB를 비롯한 하위 비트에서는 XOR 연산의 결과와 산술 뺄셈 연산의 결과가 유사하여 원본값의 비트 정보가 누출될 가능성이 있으므로, 네 번째 비트까지는 XOR 연산을 수행하지 않는다. 비교 연산의 결과는 양쪽의 비트가 다른 부분 중 상위 비트에 의해 결정되므로 오차가 발생할 확률이 낮고, 비교 연산의 오차로 인한 영향이 크지 않다. 실험 결과, 신경망의 성능에 영향을 주지 않음을 확인했다.

또한 고정된 입력에 대해 항상 같은 결과를 출력하지 않도록 중간 연산 및 출력 결과에 난수를 섞어 주되, XOR 연산만으로 1과 0을 반전시킬 수 있도록 난수의 최하위 비트를 0으로 조정한다.

5.3 마스킹된 입력값 조정 알고리즘

알고리즘 4의 비교 연산을 이용해, 튜플의 원본값을 복원하지 않고도 입력값이 임계점을 초과하지 않도록 조정하는 알고리즘을 설계할 수 있다.

먼저 입력값의 튜플과 저장되어 있는 임계점의 튜플에 비교 연산을 적용하여, 입력이 임계점을 넘지 않을 때는 그대로 x에 해당하는 튜플을, 임계점을 넘은 경우 임계점에 해당하는 튜플을 출력하도록 한다. 이때, x의 값에 따른 타이밍 차이가 발생하지 않도록 조건문을 사용하지 않고 항상 동일한 연산을 수행하도록 한다. 비교 연산의 결과는 1 또는 0에 해당하는 튜플이므로, 곱연산과 합연산을 이용해 원하는 값만 남도록 조정한다. 고정 소수점 간의 곱셈이 아니므로, 버림 연산은 진행하지 않는다.

Alg.5. Masked Fitting

Input: 2-sharing input $< x >$ and
2-sharing threshold $< a >$
Output: 2-sharing $< z >$ s.t.
$z = a$ if $x \geq a$
$z = x$ if $-a \leq x < a$
$z = -a$ if $x < -a$

1. $c := masked\ cmp(< x >, < a >)$
2. $d := masked\ cmp(< x >, < -a >)$
3. $< \neg c > := (< c >_0 \oplus 1, < c >_1)$
   $< \neg d > := (< d >_0 \oplus 1, < d >_1)$
4. $< z > = masked\ add($
   $masked\ Dot(< \neg c >, < a >),$
   $masked\ Dot(< c >, < x >))$

   $< z > = masked\ add($
   $masked\ Dot(< \neg d >, < z >),$
   $masked\ Dot(< d >, < -a >))$
5. $return\ \ < z >$

Fig. 10. Masked Input Fitting Algorithm.

5.4 마스킹된 근사 활성화 함수 알고리즘

앞선 알고리즘을 합성하여 마스킹된 활성화 함수를 설계할 수 있다. 먼저 입력값을 조정하여 만든 튜플 값을 얻고, 해당 튜플을 반복하여 곱함으로써 x의 거듭제곱에 해당하는 튜플을 계산하여 벡터 형태로 저장한 후, 이를 벡터 형태로 저장된 다항식의 계수와 내적하면 $f(x) = a_0 + a_1 x + ... + a_n x^n$에 해당하는 튜플을 얻을 수 있다. 모든 과정은 마스킹된 상태로 진행되며, 각 과정마다 출력값에 새로운 난수를 적용하므로 중간값의 원본이 복원되지 않는다.

Alg.6. Masked Activation

Input: 2-sharing input $< x >$, threshold $< a >$ and masked coefficient of $f(x)$,
$< C > = (< c_0 >, < c_1 >, ... < c_n >)$
Output: 2-sharing $< z >$ s.t. $z = f(x)$

1. $< t > := masked\ fitting(< x >, < a >)$
2. $< X_0 > := < 1 >$
3. **for** i from 0 to $n-1$ **do**
   $< X_i > := masked\ mul(< X_{i-1} >, < t >)$
4. $< X > = (< X_0 >, < X_1 >, ..., < X_n >)$
5. $< C > = (< c_0 >, < c_1 >, ..., < c_n >)$
6. $< z > := masked\ Dot(< X >, < C >)$
   $< z > = masked\ Trunc(< z >)$
7. $return\ \ < z >$

Fig. 11. Masked Approximated Activation Function Algorithm.

5.5 마스킹된 가중치 연산 및 최종 출력 연산

순방향 신경망을 구성하기 위해서는 활성화 함수 계층 이외에도 각 계층 사이에서 가중치를 적용하는 선형 연산과 출력 계층에서 최종 예측값을 결정하는 연산이 필요하다. 선형 연산은 내적과 덧셈 연산의

반복으로 구성되며, 예측 연산은 입력 조정 연산을 응용하여 알고리즘 8과 같이 설계할 수 있다. 즉, 출력 계층의 첫 번째 노드의 결과값에 해당하는 튜플을 기준으로, 다음 출력 노드의 결과값 튜플과 마스킹된 비교 연산을 수행하여 둘 중 큰 값을 새로운 기준값으로 삼고, 예측값을 노드의 번호로 업데이트한다.

Alg.7. Masked Linear Operation

Input: 2-sharing input vector $< I >^n$ and
weight matirx $< W >^{m \times n}$,
bias vector $< B >^m$
Output: 2-sharing vector $< H >$ s.t.
$H = I \cdot W + B$

1. **for** i from 0 to $m-1$ **do**
   $< H_i > := masked\ Dot(< I >, < W_i >)$
   $< H_i > = masked\ Trunc(< H_i >)$
   $< H_i > = masked\ add(< H_i >, < B_i >)$
2. $return\ < H >$

Fig. 12. Masked Linear Operation Algorithm ($< W_i >$ is $i'th$ row of matrix $< W >$).

Alg.8. Masked Prediction

Input: 2-sharing output vector $< O >^k$
Output: Argmax Label $pred$ s.t.
$O_{pred} = \max(\{O_i\})$
(O_i is output of i'th output layer node)

1. $< max > := < O_0 >, pred := < 0 >$
   $< label > = < 0 >$
2. **for** i from 0 to $n-1$ **do**
   $< label >_0 += 1$
   $c := masked\ cmp(< max >, < O_i >)$
   $< \neg c > := (< c >_0 \oplus 1, < c >_1)$
   $< z > = masked\ add($
   $masked\ Dot(< \neg c >, < max >),$
   $masked\ Dot(< c >, < O_i >))$
   $< z > = masked\ add($
   $masked\ Dot(< \neg c >, < pred >),$
   $masked\ Dot(< c >, < label >))$
3. $return\ < pred >$

Fig. 13. Masked Prediction Algorithm.

VI. 마스킹된 신경망의 안전성 및 성능 평가

5장에서 제안한 마스킹 알고리즘의 안전성 및 신경망의 성능에 미치는 영향을 평가하기 위해, 두 가지 실험을 진행하였다. 첫 번째는 해당 알고리즘이 전력 분석 공격에 대해 안전한지를 검증하기 위한 전력 누출 평가 실험이며, 상대적으로 작은 2 x 2 x 2 크기의 네트워크로 실험을 진행하였다. 두 번째는 실제 MNIST 데이터셋을 사용하여 추론 정확도를 확인하고, 마스킹을 적용하지 않은 네트워크와의 속도 및 정확도를 비교하는 실험으로, 4장과 동일한 구조의 네트워크를 사용하였다.

6.1 마스킹 알고리즘의 안전성 실험

6.1.1 전력 파형 시뮬레이터 ELMO

전력 파형 시뮬레이터 ELMO(Evaluating Leaks for the ARM Cortex-M0)는 McCann 외 [8]가 제시한 전력 파형 모델링 기법을 사용한 프로그램으로, 알고리즘 코드를 컴파일하여 생성된 어셈블리 코드를 특정 기기(ARM Cortex-M0)에서 실행시켰을 때 예상되는 전력 파형을 출력하는 시뮬레이터이다. 동일 연산에 대해 정확한 타이밍을 유지하며 노이즈가 없으므로, 적은 수의 파형으로도 정확한 결과를 도출할 수 있어 활용도가 높아지고 있다.

6.1.2 TVLA(Test Vector Leakage Assessment)

TVLA는 부채널 취약점 분석을 위해 제안된 평가 기법으로, 고정된 비밀값에 대해 고정된 입력값을 준 파형 집합과 랜덤한 입력값을 준 파형 집합이 통계적으로 다른 특성을 보인다는 점에 착안하여 두 파형 집합에 대한 Welch's t-test를 통해 취약 여부를 평가하는 fixed vs random 방식이 일반적으로 널리 사용된다. 두 파형 집합의 각 지점에 대해 다음 수식에 따라 t값을 계산하고, t의 절댓값이 4.5를 초과하는 경우 취약하다고 판단한다.

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}} \quad (2)$$

6.1.3 ELMO를 사용한 TVLA 실험

5장의 알고리즘을 바탕으로 구현한 마스킹된 순방향 신경망 알고리즘에 고정된 값과 랜덤한 값을 각각 입력하여 4,000개의 전력 파형을 ELMO로 생성한 뒤 TVLA를 진행했다.

파형의 간소화를 위해 은닉 계층은 1층으로, 입력, 출력, 은닉층의 노드 수는 각각 2개로 설정했다.

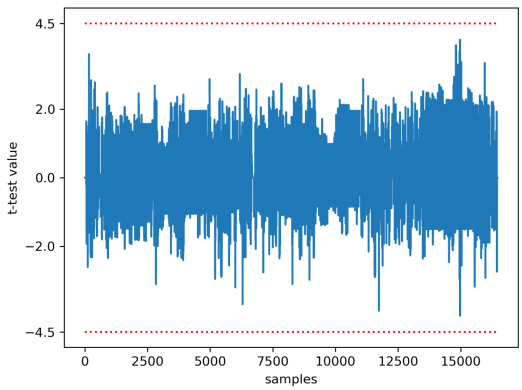


Fig. 14. TVLA test result with 4K samples of ELMO traces.

6.1.4 실제 기기 환경에서의 TVLA

ELMO를 사용한 시뮬레이션 파형뿐 아니라 실제 환경에서의 소비 전력 파형을 이용한 실험도 진행하기 위해, Chip Whisperer Lite와 STM32F4를 사용해 수집한 전력 파형을 대상으로 TVLA를 수행

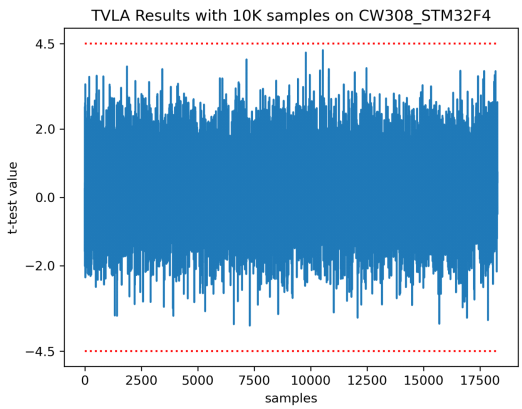


Fig. 15. TVLA test result with 10K samples of STM32F4 traces.

Table 4. Experimental Setup.

Power Board	Chip Whisperer-Lite
Target Processor	STM32F415
Sampling Rate	7.37 MS/s
Clock	7.37 MHz

했다. 실험 환경은 표 4와 같다.

ISO/IEC 17825:2016 표준 보안 레벨3의 기준에 따라 총 10,000개의 파형을 수집했으며, 6.3절의 실험과 같이 고정된 입력을 넣은 파형과 랜덤한 입력값을 넣은 파형을 각각 수집해 fixed vs random 테스트를 진행했다.

그림 6과 7에서 볼 수 있듯, 시뮬레이션 파형과 실제 파형 모두 알고리즘의 수행 과정 전체에서 신뢰도 99.99% 기준값인 절댓값 4.5를 초과하는 부분이 발견되지 않았다. 즉, 5장에서 제시한 마스킹된 알고리즘이 유효하며 해당 알고리즘으로 구성된 순방향 신경망이 전력 분석을 통한 누출에 대해 안전하다고 할 수 있다.

6.2 마스킹된 순방향 신경망의 성능 실험

6.2.1 실험 설계

5장에서 설명한 알고리즘들을 이용하면 기존의 활성화 함수 대신 다항함수 기반의 근사 함수를 사용하는 순방향 MLP 모델에 필요한 모든 연산에 마스킹을 적용하여 구성할 수 있다. 다만 고정 소수점 방식과 마스킹을 적용하여 연산하는 과정에서 필연적으로 발생하는 오차가 존재하므로, 해당 알고리즘으로 구성한 신경망의 연산이 정상적으로 작동하며 마스킹이 적용되지 않은 경우와 동등한 모델 성능을 유지할 수 있음을 확인할 필요가 있다.

따라서 4장에서 실험에 이용한 MLP 모델을 시그모이드 함수로 학습시킨 뒤, 학습된 가중치를 이용하여 입력 데이터에 대해 추론을 진행하는 순방향 신경망을 마스킹된 알고리즘으로 구현하고 추론 정확도를 확인하였다. 모델의 구조 및 하드웨어 환경은 4장의 실험과 같으며, 신경망 구현에는 C언어를 사용했다.

6.2.2 실험 결과

은닉 계층이 1계층인 모델에 대해, 부동 소수점 변수를 이용하며 시그모이드 함수를 사용하는 기존의

Table 5. Accuracy and Runtime of Feedforward Network.

Network	Accuracy	Avg Runtime(ms)
Sigmoid(float)	0.9775	0.1622
Approximated (fixed)	0.9776	0.0955
Masked network(share)	0.9774	0.3646

순방향 알고리즘과 고정 소수점 변수를 이용하며 근사 함수를 이용하는 알고리즘, 마지막으로 본 논문에서 제시한 마스킹 알고리즘을 적용한 경우의 추론 정확도 및 평균 수행 시간을 각각 측정하였으며, 이때 근사 함수로는 체비쇼프 1차 근사를 바탕으로 조정한 함수를 사용하였다. (마스킹 알고리즘은 활성화 함수 계층뿐만 아니라 추론 과정 전체에 적용하였다.)

표 5에서 볼 수 있듯이, 순방향 신경망에 마스킹을 적용하여도 높은 추론 정확도를 유지함을 알 수 있으며 기존의 활성화 함수를 사용하는 네트워크와 비교해 0.03%p 정도의 작은 차이만을 보였다.

또한 시그모이드 함수를 고정 소수점을 이용하는 근사 함수로 치환한 경우 수행 시간이 약 40% 감소했으며, 지수함수 기반의 활성화 함수를 근사하면 속도 증가에도 긍정적인 영향을 줄 수 있음을 확인했다. 마스킹을 적용하여 오버헤드가 증가하는 경우에는 원본 함수를 사용하는 네트워크에 비해 수행 시간이 약 2.2배로 증가했다.

VII. 결 론

본 논문에서는 딥러닝 신경망에 대한 부채널 공격 및 역공학의 위험성을 제시하고, 이에 대해 부채널 공격에 강인한 딥러닝 신경망 설계를 위한 방법으로 활성화 함수를 근사하여 마스킹 기법을 적용하는 새로운 방법을 제안하였다. 또한 이에 대한 성능 및 안전성 검증 실험을 수행하여 제안한 알고리즘이 비밀값에 관한 전력 정보 누출을 막을 수 있다는 것과 알고리즘을 사용한 순방향 신경망이 성능 감소 없이 정상적으로 작동한다는 것을 확인했다. 이를 통해 지수함수 기반 활성화 함수를 마스킹된 근사 함수로 대체하여 유의미한 성능 차이 없이 부채널 저항성을 높일 수 있음을 보였다.

이후에는 CNN 등 추가 연산 계층이 포함된 네트워크나 컬러 이미지로 구성된 CIFAR 등 학습이 어렵다고 알려진 데이터를 사용하는 케이스에 대한 정확도 및 부채널 분석 실험 또한 진행할 예정이다. 다만 더욱 복잡한 모델 또는 민감한 데이터를 사용하는 경우 정밀한 근사를 위해 고차 다항함수가 필요할 수 있으며, 고정 소수점 변수로 고차 다항함수를 계산하려면 더욱 큰 자료형이 필요해지는 등의 단점이 존재한다. 따라서 이후에는 부동 소수점 자료형에 대한 마스킹 기법이나 활성화 함수를 대체하는 다항식 근사 이외의 방법 등, 추가적인 연구를 계획하고 있다.

References

[1] Y. LeCun, Y. Bengjo, G. Hinton, "Deep Learning," Nature 521.7553(2015), 436-444.

[2] Chabanne, Hervé, et al, "Side channel attacks for architecture extraction of neural networks," CAAI Transactions on Intelligence Technology, 2021.

[3] Batina, Lejla, et al, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," 28th USENIX Security Symposium (USENIX Security 19). 2019.

[4] M. Méndez Real, R. Salvador, "Physical side-channel attacks on embedded neural networks: A survey," Applied Sciences, 2021.

[5] K. Athanasiou, T. Wahl, AA. Ding, , and Y. Fei, "Masking feedforward neural networks against power analysis attacks," Proceedings on Privacy Enhancing Technologies, 2022(1).

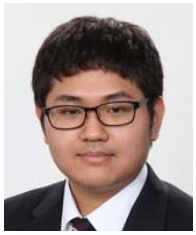
[6] N. G. Timmons, A. Rice, "Approximating Activation Functions," arXiv preprint arXiv:2001.06370, 2020.

[7] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," In 2017 IEEE Symposium on Security and Privacy (SP), pages 19 - 38. IEEE, 2017.

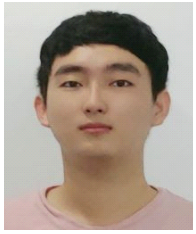
〔8〕 W. Shin, S. Jung, H. Kim, "Side-Channel Countermeasure for Binary Neural Network Protection," Conference on Information Security and Cryptography Summer 2023, pp.75, 2023.

〔9〕 McCann, David; Oswald, Elisabeth; Whitnall, Carolyn, "Towards Practical Tools for Side Channel Aware Software Engineering:'Grey Box'Modelling for Instruction Leakages," In: 26th USENIX security symposium (USENIX security 17), pp. 199-216. 2017.

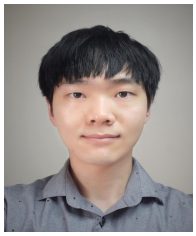
〈저자소개〉



김 준 섭 (Joonsup Kim) 학생회원
2017년 2월: 고려대학교 수학과 학사
2023년 3월~현재: 고려대학교 정보보호학과 석사과정
〈관심분야〉 부채널 분석, 딥러닝 네트워크 보안



김 규 상 (GyuSang Kim) 학생회원
2020년 2월: 연세대학교 수학과 학사
2020년 9월~현재: 고려대학교 정보보호학과 석박사 통합 과정
〈관심분야〉 공개키 암호, 부채널 공격



박 동 준 (Dongjun Park) 학생회원
2018년 8월: 세종대학교 정보보호학과 학사
2020년 8월: 고려대학교 정보보호학과 석사
2020년 9월~현재: 고려대학교 정보보호학과 박사 과정
〈관심분야〉 부채널 공격

박 수 진 (Sujin Park) 학생회원
2023년 2월: 고려대학교 과학기술대학 인공지능사이버보안학과 학사
2023년 3월~현재: 고려대학교 정보보호대학원 정보보호학과 석사과정
〈관심분야〉 부채널 공격, 부채널 대응기술, 암호시스템 안전성 분석

김 희 석 (HeeSeok Kim) 종신회원
2006년: 연세대학교 수학과 학사
2008년: 고려대학교 정보보호대학원 석사
2011년: 고려대학교 정보보호대학원 박사
2011년 9월~2012년 12월: Bristol University 박사후연구원
2013년~2016년 8월: 한국과학기술정보연구원(KISTI) 선임연구원
2015년~2016년 8월: 과학기술연합대학원대학교(UST) 조교수
2016년 9월~현재: 고려대학교 과학기술대학 인공지능사이버보안학과 부교수
〈관심분야〉 부채널 공격, 암호시스템 안전성 분석 및 고속구현, 암호칩 설계 기술, 보안관제, 네트워크 보안

홍 석 희 (Seokhie Hong) 종신회원
2001년: 고려대학교 수학과 박사
1999년 8월~2004년 2월: ㈜시큐리티 테크놀리지 선임연구원
2003년 3월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U. Leuven ESAT/SCD-COSIC 박사후 연구원
2005년 3월~2013년 8월: 고려대학교 정보보호대학원 부교수
2013년 9월~현재: 고려대학교 정보보호대학원 정교수
〈관심분야〉 대칭키 및 공개키 암호 알고리즘, 부채널 공격 및 대응기법, 디지털 포렌식