Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.3, Jun. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.3.487

TPMP : ARM TrustZone을 활용한 DNN 추론 과정의 기밀성 보장 기술*

송 수 현,[1†] 박 성 환,[1] 권 동 현[2‡]
[1,2]부산대학교 (대학원생, 교수)

TPMP: A Privacy-Preserving Technique for DNN Prediction Using ARM TrustZone*

Suhyeon Song,[1†] Seonghwan Park,[1] Donghyun Kwon[2‡]
[1,2]Pusan National University (Graduate student, Professor)

요 약

딥러닝과 같은 기계학습 기술은 최근에 광범위하게 활용되고 있다. 이러한 딥러닝은 최근 낮은 컴퓨팅 성능을 가지는 임베디드 기기 및 엣지 디바이스에서 보안성 향상을 위해 ARM TrustZone과 같은 신뢰 수행 환경에서 수행되는데, 이와 같은 실행 환경에서는 제한된 컴퓨팅 자원으로 인해 정상적인 수행에 방해를 받는다. 이를 극복하기 위해 DNN 모델 partitioning을 통해 TEE의 제한된 memory를 효율적으로 사용하며 DNN 모델을 보호하는 TPMP를 제안한다. TPMP는 최적화된 memory 스케줄링을 통해 기존의 memory 스케줄링 방법으로 수행할 수 없었던 모델들을 TEE 내에서 수행하여 시스템 자원 소모를 거의 증가시키지 않으면서 DNN의 높은 기밀성을 달성한다.

ABSTRACT

Machine learning such as deep learning have been widely used in recent years. Recently deep learning is performed in a trusted execution environment such as ARM TrustZone to improve security in edge devices and embedded devices with low computing resource. To mitigate this problem, we propose TPMP that efficiently uses the limited memory of TEE through DNN model partitioning. TPMP achieves high confidentiality of DNN by performing DNN models that could not be run with existing memory scheduling methods in TEE through optimized memory scheduling. TPMP required a similar amount of computational resources to previous methodologies.

Keywords: Arm TrustZone, Deep Learning, model privacy

I. 서 론

딥러닝(Deep Learning)은 기계 학습(Machine Learning)의 한 가지 기술로서 이미지 분류, 음성 인식, 얼굴 인식, 의료 진단 등에서 광범위하게 사용되고 있다. 특히, 최근에는 클라우드 서버뿐만 아니라 모바일 기기나 사물 인터넷 기기와 같은 엣지 디바이스와 임베디드 기기에서 이러한 딥러닝 연산을 수행하는 경우가 점차 늘어나고 있다[1][2][3][4]. 예를 들어, 모바일 기기에서 얼굴, 음성, 지문 등의 인식이나 멀웨어 탐지에 딥러닝이 활용되고 있다[5][6][7]. 하지만 이처럼 딥러닝이 다양한 분야에 활용되면서 이러한 딥러닝 연산의 핵심이라고 할 수 있

Received(05. 03. 2022), Modified(05. 27. 2022), Accepted(05. 30. 2022)
* 본 연구는 2020학년도 부산대학교 교내학술연구비(신임교수 연구정착금)에 의한 연구임.
† 주저자, sshpnu@pusan.ac.kr
‡ 교신저자, kwondh@pusan.ac.kr(Corresponding author)

는 DNN 모델 정보에 대한 공격 시도들도 늘어나고 있다[8]. DNN(Deep Neural Network)모델 정보란 DNN의 계층 정보, 가중치와 같은 매개변수를 의미한다. DNN 모델에 대한 공격은 이러한 정보들을 대상으로 하며 그 공격 중에는 model extraction 공격이 있다. model extraction 공격은 타겟 모델과 유사한 성능을 가지는 유사 대체 모델을 생성하는 것을 목표로 하거나, 이를 위해 타겟 모델의 함수 매개변수, 활성 함수, 최적화 알고리듬, 계층의 개수 등과 같은 정보를 알아내는 것에 중점을 둔다.

따라서 최근에는 이러한 공격에 대응하여 딥러닝 연산 중 DNN 모델의 기밀성(confidentiality 혹은 privacy)을 보호하는 다양한 연구들에 제안되고 있다[9][10]. 그 중 DarkneTZ[11]는 컴퓨팅 자원이 부족한 임베디드 기기에서 ARM TrustZone[13]이라고 하는 신뢰 수행 환경(Trusted Execution Environment, TEE)를 활용해 DNN 모델을 보호하는 기술을 제안하였다. 즉, 딥러닝 연산을 TEE에서 수행함으로써 일반 수행 환경(Rich Execution Environment, REE)에 존재하는 취약점 및 공격으로부터 DNN 모델을 비롯한 모델의 매개변수를 보호할 수 있기 때문이다. 하지만 해당 논문의 소스 코드[14]를 분석한 결과 DarkneTZ는 TEE의 제한된 memory 크기를 고려하지 않은 설계로 제한된 TEE memory상에서 보호할 수 있는 계층의 개수가 적으며 이는 보호되는 매개변수가 적다는 것을 의미한다.

이에 본 논문에서는 ARM TrustZone을 활용해 임베디드 디바이스에서 딥러닝 추론 연산의 기밀성을 효율적으로 보장하는 TPMP를 제안한다. TPMP는 크게 TPMP Manager와 TPMP Migrator로 이루어져 있다. 먼저 TPMP Manager는 주어진 DNN 모델의 크기와 사용 가능한 TEE memory양을 토대로 최적의 딥러닝 연산 분할(partition)을 얻고 이를 토대로 딥러닝 연산을 REE와 TEE에서 분할하여 수행하도록 하는 역할을 담당한다. TPMP Migrator는 딥러닝 연산 중 일부를 TEE에서 수행하는 경우 적은 TEE memory로 연산을 수행할 수 있도록 전처리 및 후처리를 담당한다. 실험 결과 TPMP는 동일한 TEE memory 크기를 사용해 DarkneTZ 대비해 DNN 모델의 더 많은 부분을 보호할 수 있음을 확인하였다. TPMP는 DarkneTZ와 동일한 환경에서 수행할 경우 수행 시간상 1% 미만의 오버헤드를 가진다.

II. 배경 지식

2.1 Deep Neural Network (DNN)

딥러닝은 기계학습의 한 종류로서 여러 단계의 학습을 통해 점진적으로 특징을 추출하여 예측값을 계산한다. 딥러닝은 크게 학습(training) 과정과 추론(inference) 과정으로 나눌 수 있다. 학습 과정에서는 학습을 위한 데이터들을 사용하여 DNN 모델을 생성하는 단계이고, 추론 단계는 학습 단계에서 생성한 DNN 모델을 통해 입력값에 대한 추론 값을 도출하는 단계이다.

이때 DNN 모델이란 딥러닝의 다양한 알고리즘 가운데 하나로 기본적으로 입력층(input layer)과 출력층(output layer) 사이에 여러 개의 은닉층(hidden layer)을 가지는 구조로 되어있다. 이때 DNN은 인간 두뇌의 신경망 동작 과정을 모방한 방식으로 추론 단계를 수행하게 된다. 즉, 각 계층(layer)에서는 주어진 입력값에 학습 과정에서 얻어진 가중치(weight)와 바이어스(bias)를 함께 연산하여 출력값을 얻고 각 계층의 출력값은 다음 계층의 입력으로 사용되어 계층 단위로 연쇄적인 연산이 이루어지게 된다. DNN 모델 정보에는 어떠한 활성화 함수 계층들로 구성이 되는지, 각 계층의 매개변수(parameter)인 입력값(input), 출력값(output), 가중치(weights), 바이어스(bias), 그리고 그 밖의 연산 수행과 관련된 설정값(configuration file)을 포함한 정보이다.

2.2 ARM TrustZone

ARM TrustZone은 엣지 디바이스에서 널리 사용되는 프로세서 아키텍처인 ARM에서 제공하는 하드웨어 기반 신뢰 수행 환경 기술이다[13]. ARM TrustZone은 memory, I/O 디바이스 등의 시스템 자원을 Normal World 혹은 Secure World를 위한 자원으로 구분하고, CPU가 Normal mode인지 secure mode 인지를 통해 각 시스템 자원에 대한 접근제어 기능을 제공한다. 즉, 각종 보안상 중요 연산을 Secure World, 즉 TEE에서 상에서 수행하게 되면 Normal World인 REE에서 동작하는 일반 연산들은 보안 연산 과정에 영향을 미칠 수 없게 되어 안전한 보안 연산을 보장할 수 있다. 한편 Secure World(TEE)와 Normal World(REE)

는 각각 독립적인 운영체제를 가질 수 있으며 World Switch를 위해서는 SMC(Secure Monitor Call)이라고 하는 특수한 명령어를 수행해야 한다.

2.3 Model extraction attack

Model extraction attack은 타겟 모델 f를 공격하여 타겟 모델과 가장 유사한 동작을 수행하는 f'을 생성하거나 타겟 모델의 정보(매개변수, 구조)를 추출하기 위해 공격자가 수행하는 공격의 일종이다. 타겟 모델과 유사한 모델 f'을 만드는 경우 입력값에 대한 출력값을 지속적으로 분석하여 모델 f와 가장 유사한 동작을 하도록 모델 f'을 지속적으로 fine tuning 한다. 또한, Model extraction을 통해 DNN 모델의 매개변수를 훔치는 목적을 가지는 경우도 있다. 이러한 모델 매개변수를 유출하는 것은 DNN 모델 알고리즘의 기밀성을 손상시키고 학습자의 지적 재산을 훔치는 것이다. 공격자는 Model extraction attack에서 유출한 매개변수와 DNN의 구조 정보를 사용하여 model inversion attacks[15] 또는 evasion attack[16]과 같은 추가적인 공격 또한 수행할 수 있다.

2.4 Gray-box and Black box attack

Model extraction attack은 adversarial knowledge의 양에 따라 공격을 Black-box attack과 Gray-box attack으로 분류한다. 타겟 모델의 부분적인 knowledge 을 가지고 있을 때, 수행되는 공격을 Gray-box model extraction attack이라고 한다. adversarial knowledge에는 타겟 모델의 매개변수, 계층 정보, 활성 함수 정보, 가중치, 입력값, 출력값의 정보들이 포함된다. 이전 연구[17]에서는 모델에 대한 adversarial knowledge를 사용하여 타겟 AI 모델의 가중치 값을 정확하게 알아낼 수 있었다. black-box attack의 경우 타겟 모델에 대해 어떠한 knowledge도 가지고 있지 않을 때 수행되는 공격을 말한다. 이 경우 공격자는 출력값을 얻기 위해 오직 입력값만을 모델에 적용할 수 있으며, 이 입력값 또한 공격자가 직접 생성하여야 한다.

공격자는 이러한 adversarial knowledge의 양에 따라 공격의 수행 접근 수준이 달라지게된다.[18] Black-box의 경우 DNN 모델의 정보가 없다고 가정하기에 모델의 설정값 또는 이전에 사용한 입력값만을 사용하여 모델의 취약성을 파악 후, 일련의 공격을 수행하여야 한다. Gray-box의 경우, 타겟 모델의 정보들을 활용하여 모델이 취약할 수 있는 기능, 오류 지점을 식별할 수 있다. 따라서 이러한 gray-box 공격을 통한 모델 정보 탈취와 같은 model extaction attack은 매우 적대적인 공격이다.

III. 위협 모델

본 논문에서는 DNN 모델의 위협은 크게 배포단계와 수행 단계로 나눈다. 배포단계에서의 위협은 엣지 디바이스로 모델 정보와 가중치가 배포될 때를 가정한다. 수행 단계는 입력값, 가중치, 매개변수등을 사용하여 연산을 수행하는 과정을 가정한다. 또한 우리는 ARM TrustZone이 탑재된 엣지 디바이스 또는 임베디드 시스템을 사용하는 것을 전제로 한다. 그리고 secure world 내에서 동작하는 코드와 데이터는 secure boot와 같은 메커니즘을 통해 안전하게 초기화되고 ARM TrustZone을 활용하여 normal world로부터 무결성 및 기밀성이 보호된다고 가정한다. 반대로 공격자는 엣지 디바이스 또는 임베디드 시스템의 normal world 환경에 구동되는 소프트웨어의 취약점을 악용할 수 있다. 또한 normal world에서 수행되는 기존 DNN연산의 경우 배포되는 가중치와 모델 데이터에 평문으로 접근, 수정이 가능하다. 이는 공격자가 adversarial knowledge에 해당하는 모델 구조, 매개변수, 가중치에 공격자가 쉽게 접근할 수 있다.

배포단계에서 매개변수와 가중치를 유출하거나 사전에 변조하여 모델의 정상적인 수행을 방해할 수 있다. 또한 DNN 연산 수행 단계에서 소프트웨어의 취약점[19][20]을 통해 연산 도중 adversarial knowledge에 접근 또는 정상적인 수행을 방해할 수 있다. 이러한 연산을 TEE내에서 수행하면 연산의 수정과 방해로부터 안전하게 수행할 수 있다. 공격자가 접근 가능한 모델 구조, 매개변수와 가중치는 모두 adversarial knowledge중의 하나로써 이를 통해 Adversary가 DNN을 simulation할 수 있는 Gray-Box Attack을 시도할 수 있다. Adversarial knowledge의 양에 따라 Model extraction attack의 성공에 큰 영향을 미친다.

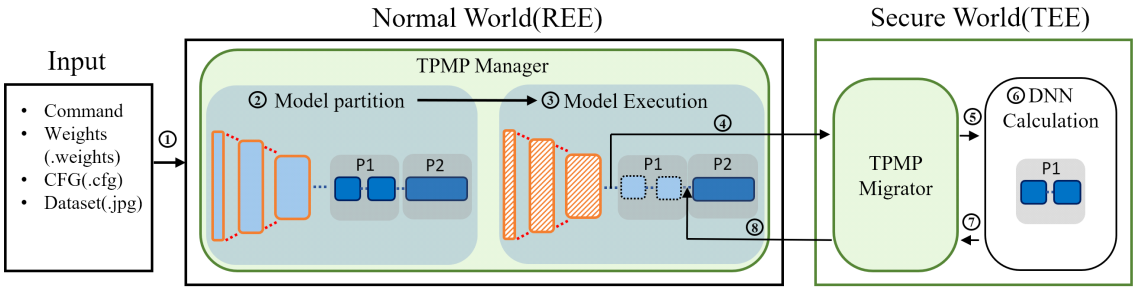


Fig. 1. A diagram of the TPMP framework.

따라서 TPMP는 DNN 연산을 secure world 내부에서 수행하여 이러한 DNN 모델 수행의 기밀성을 보장하고자 한다.

기존 연구[11]들에서 나와 있듯 DNN 모델에 대한 유출 공격들은 후반부 계층들의 정보에 집중되므로 TPMP도 마찬가지로 후반부 계층들을 보호의 대상으로 삼는다. 또한 TPMP는 미리 학습된 모델과 암호화된 가중치를 사용하여 추론 연산만을 TEE 내에서 수행하는 것을 전제로 한다. 전반 계층은 입력값의 크기를 줄이는 과정으로, 전반 계층의 매개변수의 크기는 전체 가중치에서 차지하는 크기가 작다. 후반 계층의 대부분은 Fully Connected 계층으로 이루어져 있으며 해당 계층의 매개변수가 전체 가중치의 대부분을 차지한다.

IV. 디자인

4.1 TPMP 개요

모든 연산이 REE에서 수행되는 기존의 환경에서는 위협 모델에 언급된 것처럼 계층의 간의 매개변수 즉 가중치 정보가 배포단계에서 보호를 받을 수 없다. TPMP는 배포단계에서, 모델의 가중치와 TEE 내에서 수행되는 계층의 정보를 Secure OS에서 제공하는 암호화 API를 기반으로 128bit AES-GCM 대칭키 암호화를 통하여 무결성을 보장한다. 또한, 수행 단계에서 최대한 많은 계층을 TEE에서 수행하여 공격자가 해당 계층이 사용하는 매개변수 및 가중치에 접근할 수 없게 하고, DNN 모델 수행 전반에 연산 변조 방지 및 계층의 무결성을 보장함으로써 공격자가 REE에서 연산에 개입하는 위험성을 크게 줄일 수 있다.

Secure world 내에서 가장 많은 양의 매개변수와 가중치를 보호할 수 있도록 TPMP는 마지막 계층부터 역순으로 계층을 보호한다. 마지막 계층부터 보호되기에, adversarial knowledge에 포함되는 출력값(e.g,. confidential score 등) 또한 REE에서 접근할 수 없도록 마스킹 된다. 따라서 출력값 분석을 통해 입력값 유추하는 MIA(Membership inference attack)[12]와 같은 공격을 수행할 수 없기에 입력값과 출력값이 공격자에게 유출되지 않는다.

이전 연구인 DarkneTZ에서는 TEE의 secure memory를 충분히 효율적으로 사용하지 않았으며, 대부분 상황에서 마지막 계층만을 TEE에서 수행하여 최종 출력값만을 가린다. 따라서 TPMP는 memory 사용방식을 개선하여 출력값뿐만 아니라 매개변수까지 보호하기 위해 더 많은 계층을 TEE 내에서 수행하도록 한다.

4.2 TPMP 동작 과정

TPMP는 계층 단위로 DNN 연산을 수행한다. 이는 DNN의 기본 구조를 해치지 않기 때문에 정확도에 영향을 미치지 않는다. 또한, TPMP는 TEE 내에 필요한 계층만을 만들어 연산을 진행한다. 이렇게 REE 또는 TEE에서 수행될 계층의 집합을 partition이라고 정의한다. 따라서 하나의 추론 연산 과정을 여러 개의 partition에 나누어 수행하므로 TEE secure memory를 효율적으로 사용할 수 있게 된다. 따라서 많은 수의 계층을 가지는 모델이라도 partition으로 분리하고, TEE를 사용하여 보호할 수 있다. 동시에 REE에 연산 되는 계층이 적어지므로 Attack Surface 또한 줄일 수 있다.

TPMP는 Fig. 1에 있는 TPMP Manager와 TPMP Migrator와 같이 크게 두 가지 모듈로 구성되어있다. TPMP Manager는 입력값에서 읽어들인 계층의 size를 근거로 TEE 내에서 연산할

strategy를 만들어낸다(Fig. 1 ①,②). strategy는 하나의 partition에 포함되는 계층의 개수, 첫 번째 및 마지막 계층 번호를 지정하며, 여러 partition의 집합이다. 만들어진 strategy에 따라 DNN 연산을 수행하는데③, 수행 중인 계층(Ln)이 strategy에 포함된다면 TPMP Migrator를 통해 해당 계층을 TEE로 전달한다④. TPMP Migrator는 TEE 내에서 DNN 모델 연산 수행을 위하여 전처리⑤와 DNN 연산⑥이 종료된 후 Normal World로 출력값을 전송하기 위한 후처리⑦를 수행한다. TPMP Migrator의 전처리⑤는 Manager에서 생성된 strategy를 기반으로 각 계층을 생성하고 DNN 연산⑥을 수행한다. 또한, 해당 partition의 연산이 끝난 이후 secure memory 상에서 partition의 마지막 계층의 매개변수 이외의 나머지 계층의 매개변수중 불필요한 정보를 모두 삭제하여 추가 memory 공간을 확보한다. Partition의 마지막 계층 매개변수는 다음 계층 또는 출력값 출력에 사용된다. 이후 Migrator의 동작이 종료되면 다음 계층이자 다음 partition의 직전의 REE 위치로 이동한다⑧. 이를 통해, TEE내에 최소한의 계층만 남겨둘 수 있다. 이러한 TPMP의 동작 개요는 Fig. 1에 나타나 있다. 각 모듈에 대한 보다 자세한 역할에 대해서는 TPMP Manager는 5.1에서, TPMP Migrator는 5.2에서 설명 한다.

V. 구 현

5.1 TPMP Manager

구체적으로 TPMP Manager는 2가지 역할, DNN Model partition, DNN Model Execution을 수행한다. DNN Model partition은 DNN 모델 분석과 strategy 생성하며 DNN Model Execution은 DNN 전체 연산의 흐름을 관리하며 strategy에 따라 TEE의 TPMP Migrator를 호출하여 연산을 수행하도록 한다.

5.1.1 DNN Model partition

Model partition 과정에서는 사전 학습된 DNN 모델의 configuration file을 파싱(parsing)하여 DNN 모델의 각 계층의 크기를 도출한다(Fig.2의 line 18). DNN 모델 Configuration File은 DNN 계층의 종류, 개수, 필터의 매개변수와 연산 수행 횟수를 포함한다. 이때, DNN의 연산이 진행되는 순차적으로 각 계층(L[n])을 스캔한다. 만약 입력값으로 받은 TEE_start에 해당하지 않는 계층이 있다면 해당 계층(L[n]) partition 번호를 0으로 지정한다.(Fig. 2 line 23) partition 번호(e)가 0으로 지정된 모든 계층은 REE에서 수행된다. 이후 partition 번호(e)가 1 이상인 계층은 모두 TEE에서 수행되며 계층 partition 번호(e)가 바뀜에 따라 TEE내에서 수행되는 partition의 순서가 달라진다. 현재 partition에서 수행될 계층의 크기(sum)가 TEE의 secure memory 크기보다 작다면 L[n]은 현재 partition(e)에 할당된다(Fig. 2 line 28). 이때, 동일 partition에 이전의 계층(L[n-1], [Ln-2] …)가 존재한다면 sum =

```
1  Initialization:
2  Total number of layer: N
3  Array of the layer: L
4  Number of the first layer in TEE: TEE_start
5  Total memory size of TEE: TEE_Memory_Size
6  Current layer index: n
7  Current partition index: e
8  Memory size of the partition: sum
9
10 Input: ModelConfigurationFile, TEE_start, TEE_Memory_Size
11 Output: finalized partitioning strategy of all layer
12
13 DNN Model Partition
14 e = 1
15 sum = 0
16
17 % make DNN
18 Parsing CFG(ModelConfigurationFile)
19
20 for n=0 to N-1
21 % Partition_Number = 0 means Calculated in REE
22     if n < TEE_Start
23         L[n] → Partition_Number = 0
24
25 % Calculated in TEE
26     else
27        sum += n → Layer_Size
28          if sum < TEE_Memory_Size
29             L[n] → Partition_Number = e
30
31 % Partition defined
32         else
33             sum = L[n-1] → Layer_Size
34             sum += L[n] → Layer_Size
35             L[n] → Partition_Number = ++e
```

Fig. 2. Algorithm of TPMP Manager

```
1  DNN Model Execution
2
3  for n=0 to N-1
4      if L[n] -> Partition_Number = 0
5          % Partition 0 will be calculated in REE
6
7      if L[n] -> Partition_Number > 0
8          n += MIGRATOR(L[n] -> Partition_Number)
9          % Remaining partition will be calculated in TEE
```

Fig. 3. Algorithm of DNN Model Execution

L[n-2] + L[n-1] + L[n] 이다. 이렇게 하나의 partition를 구성하는 과정은 partition에 포함되는 계층의 memory 크기의 합(sum)이 TEE의 secure memory보다 커질 때까지 반복된다. sum이 TEE secure memory보다 커지면(Fig. 2 line 32) 다음 partition를 만든다. 다음 partition의 경우 초기 sum의 크기를 L[n-1]의 크기로 초기화한다. 새로운 partition에서 연산이 수행되는 경우, 현재 계층의 이전 계층(L[n-1])의 output을 참고하여 수행하기 때문에, secure memory 상에 해당 매개변수가 존재할 수 있는 영역을 할당해 줄 필요가 있다.(Fig. 2 line 33) 이후 같은 방식으로 partition를 생성한다. 이렇게 생성되는 배열 L[N]을 strategy로 정의한다.

5.1.2 DNN Model Execution

Model Execution은 DNN의 연산 흐름을 관리한다. 5.1.1장에서 생성된 strategy(L[N])를 기반으로 DNN을 생성한다. 이렇게 생성된 DNN은 계층 순서대로 수행되며 현재 계층(L[n])의 partition 번호가 0 인경우에서는 REE에서(Fig. 3 line 4), 1 이상인 경우 현재 계층(L[n])을 TPMP Migrator에 전달하여 TEE에서 수행되도록 한다(Fig. 3 line 7). partition에 포함된 모든 계층이 수행이 완료된 경우, strategy를 참조하여 모든 DNN 연산을 종료하고 결과를 출력할 것인지, 다음 계층(L[n+1])을 TEE로 전달하여 다음 partition를 수행할지 결정한다. 이러한 DNN Model Execution 동작 과정들은 수행시간(run-time)에 수행되며 각 계층을 하나씩 스캔하며 동작한다.

5.2 TPMP Migrator

Migrator는 DNN의 연산을 TEE에서 수행할 수 있도록 매개변수들의 전처리를 수행하고 TEE 내에서의 DNN 연산 이후 후처리를 수행하여, 총 두 가지의 과정을 거친다. 전처리(Fig. 4 line 7)는 TEE에서 계층의 연산을 수행할 수 있도록 계층을 활성화한다. TPMP Manager로부터 전달받은 strategy(L[N])를 토대로 strategy에 저장된 계층의 순서와 개수에 따라 계층의 outline을 생성(Fig. 4 line 9)한다. 계층의 가중치는 암호화되어 있으므로 전처리 과정에서 복호화(Fig. 4 line 10)한다. 이렇게 TEE에서 계층과 가중치를 활성화하는 과정을 전처리로 정의한다. 이렇게 활성화된 계층을 calculation에 전달한다. calculation은 TPMP Migrator의 전처리를 통해 계층이 활성화된 이후 DNN 연산을 수행(Fig. 4 line 17)하는 과정이다. 계층의 outline이 올바르게 만들어져 있어야 정확하게 DNN의 연산을 진행할 수 있으며 모든 calculation 과정은 TEE에서 수행된다. 이후 calculation 과정을 통해 연산의 출력값을 생성한다. 만약 해당 partition의 마지막 계층이 전체 DNN 연산의 마지막 계층이면 후처리(Fig. 4 line 20)를 통해 DNN 연산의 출력값을 다시 REE에 전달한다. 이렇게 strategy가 끝나면 TEE에 남아 있는 활성화된 계층을 삭제(Fig. 4 line 21)한다. 만약 다음 partition이 존재한다면 현재의 partition에 존재하는 마지막 계층을 제외한 모든 계층의 매개변수들을 삭제한다. 마지막 계층은 다음 partition에 포함되며, 다음 partition의 연산이 시작될 때 이전 partition의 마지막 계층 매개변수들이 삭제된다.

```
1  Initialization:
2  Array of the weights: W
3  Array of the outputs: O
4
5  MIGRATOR
6
7  % PREFIXION, make DNN and decrypt Weights in TEE
8  while n <= last layer of partition e
9       Parsing CFG(n, ModelConfigurationFile)
10      TEE ← load W[n] & decrypt W[n]
11      n++
12
13
14 for n to last layer of current partition e
15
16 % CALCULATION, calculate layer L[n] in tee
17      forward network calculation(L[n])
18      return O[n]
19
20 % POSTFIXION, free used layer, return the outputs
21      free layer TEE(L[n], … , L[last layer of partition-1])
22
23      return REE ← O
24
25 % return REE and proceed next partition or end the DNN
```

Fig. 4. Algorithm of TPMP Migrator

VI. 실험 결과

TPMP가 DNN 연산을 수행할 때 발생하는 overhead와 계층 활용 효율성을 평가한다. TPMP와 기존의 ARM TrustZone을 활용하는

DarkneTZ 모델을 사용하며 Fine-Tuning 된 AlexNet(Fig. 5)에 대한 추론을 수행한다. TPMP와 DarkneTZ는 동일한 설정에서 DNN을 수행한다. TEE 내에서 연산 되는 계층 개 수별 CPU 실행시간 및 memory 사용량을 측정한다. 마지막 계층부터 TrustZone이 수행 가능한 대부분 계층을 포함하며, 이에는 Fully Connected 계층, SoftMax 계층, Maxpool 계층이 포함된다. 기존 DarkneTZ의 비효율적인 memory 사용을 개선한 TPMP의 효율성을 보여주고, 이로 인해 발생하는 CPU 실행시간 overhead의 수치를 제시한다. Open portable TEE(OP-TEE)를 구현하기 위해 1.4Ghz로 동작하는 ARM Cortex-A53와 1GB SRAM으로 구성된 Raspberry Pi 3 B+를 사용한다. TEE secure memory의 크기는 TEE 런타임이 TA에 memory를 할당할 때 설정된 크기로 고정되며 TPMP의 경우 memory 효율성 및 사용량 평가를 위해 각각 4MiB와 3MiB로 제한하여 추론을 수행한다. DarkneTZ의 경우 TEE secure memory는 4MiB로 제한한다.

실험 시 추론에 사용한 모델인 AlexNet은 5개의 Convolution 계층과 3개의 Fully Connected 계층으로 이루어져 있다. 추론에 입력으로 사용되는 데이터셋은 CIFAR-10 (Fig. 6) [21] 을 사용하였다. 이 데이터 세트는 32x32 크기의 60000개의 이미지로 이루어져 있다. 또한, 10개의 클래스로 분류된다. 따라서 클래스마다 6000개의 이미지로 구성된다. 이 중, 50000개의 이미지는 학습에 사용되는 학습데이터이며 나머지 10000개의 이미지는 추론에 사용되는 데이터이다.

DNN	Architecture
AlexNet	C96-MP-C256-MP-C384-MP-C256-MP-FC512-D0.5-FC512-D0.5-FC512-SM

Fig. 5. Architecture notation: Convolution layer(C):the number of filters, 3x3 of filter size. Fully connected(FC): the number of neurons(output). Dropout layer(D): dropping rate

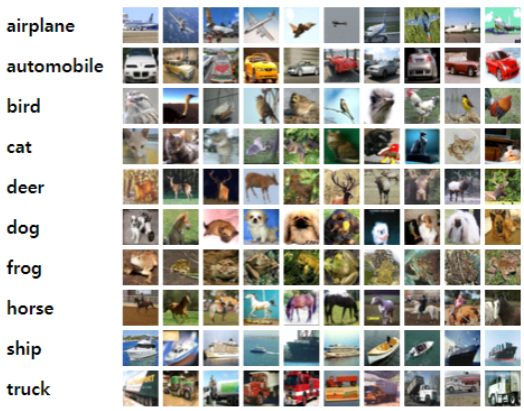


Fig. 6. Cifar-10 Dataset [21]

6.1 Memory 사용량

Fig. 7은 추론시 DNN의 총 memory 사용량이다. Fine-tuned된 AlexNet을 사용하였으며 x축은 TEE 내에서 수행되는 계층의 개수이며, 이는 마지막 계층부터 합산된 개수이다. 이 AlexNet에서는 13개의 계층이 수행되며, TEE에서 6개의 계층이 수행된다면 8번부터 13번까지의 계층을 계산한다는 의미이다. Fig. 8 에서 볼 수 있듯이, TPMP는 DarkneTZ에서 수행할 수 없는 memory 사용량이 큰 모델인 Fine-Tuned AlexNet의 계층 4개 이상을 TEE에서 성공적으로 수행한다. DarkneTZ의 경우 TEE 내에서 연산 되는 모든 계층의 memory 사용량의 총합(sum)이 DarkneTZ의 TEE secure memory 크기인 4MiB를 넘는 경우인 TEE 계층 3개 이상의 시나리오에서는 추론을 수행할 수 없었다. TPMP의 경우 TEE내의 계층들의 memory 사용량의 총합이 secure memory 크기보다 크더라도 Analyzer의 strategy에 따라 계층을 partitioning 하여 성공적으로 추론을 마쳤다.

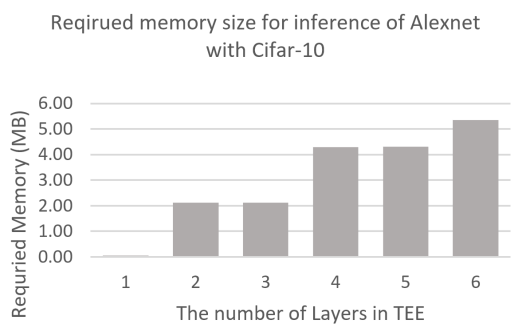


Fig. 7. Total required memory consumption of AlexNet inference

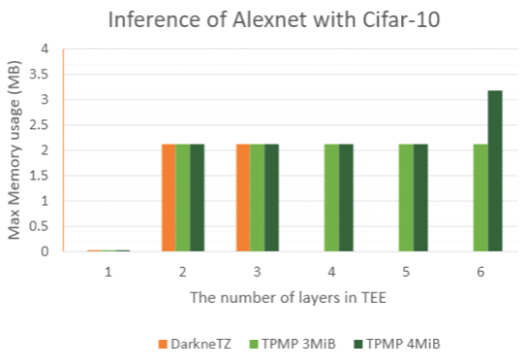


Fig. 8. Maximum memory usage of AlexNet inference

TEE에서 6개의 계층을 연산하면 총 memory 사용량이 5MiB 이상이지만 TPMP는 계층을 분할하여 각 partition은 TEE secure memory 4MiB 중 최대 3.2MiB, 3MiB 중 최대 2.2MiB만 사용한다. TPMP는 DarkneTZ에 비해 계층당 8Bytes를 추가로 사용한다. partition에서 사용되는 계층만 동적으로 할당되면서 추가로 요구되는 구조체를 위한 memory overhead이다.

이렇게 TPMP는 추론 시, 계층 partitioning을 통해 더 많은 계층을 TEE 내에서 수행, 보호하면서 memory overhead를 최소화하여 기존 DarkneTZ와 큰 차이를 나타내지 않는다.

위 Fig. 9과 10은 TPMP의 TEE 내에서 수행되는 계층이 총 6개인 시나리오에서 TEE Secure memory 크기에 따라 partitioning이 다르게 적용되는 경우를 보여준다. Fully Connected 계층 두 개를 수행하는데 요구되는 memory size는 약 3MiB 이상이며, Fig. 9 같이 TEE secure memory가 3MiB인 경우, 두 개의 Fully Connected 계층을 각각 다른 partition에서 수행하였다.

Fig. 10과 같이 4MiB인 경우는 두 개의 Fully Connected 계층을 하나의 partition에서 수행했다. TEE 내에 수행되는 계층의 수가 1개부터 3개인 경우, TEE 내의 모든 계층이 수행되는데 필요한 memory가 TEE Secure memory보다 작으므로 모든 계층을 하나의 partition에서 수행하였다. 또한, TEE 내에서 수행되는 계층의 수가 4개와 5개인 경우에는 약 4.2MiB의 크기를 가지는 계층들을 두 개의 partition으로 나누어 수행하였으며, 이는 TEE secure memory의 크기가 3MiB와 4MiB로 다른 경우에도 차이는 없다.

6.2 CPU 수행시간

CPU 수행시간은 CPU가 instruction을 수행하

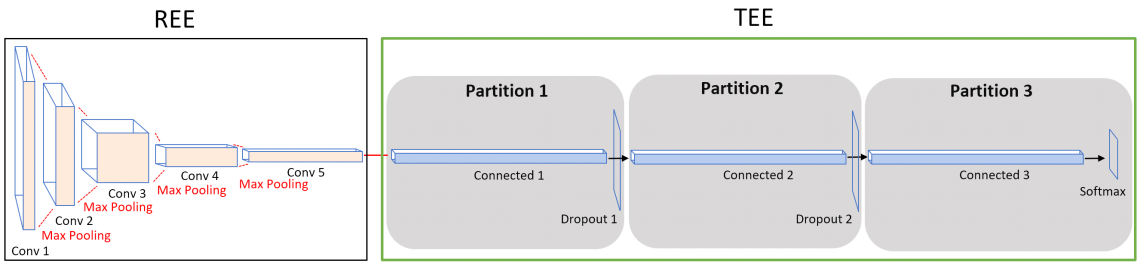


Fig. 9. Alexnet partitioned into 3 partitions with 3MiB secure memory for each partition.

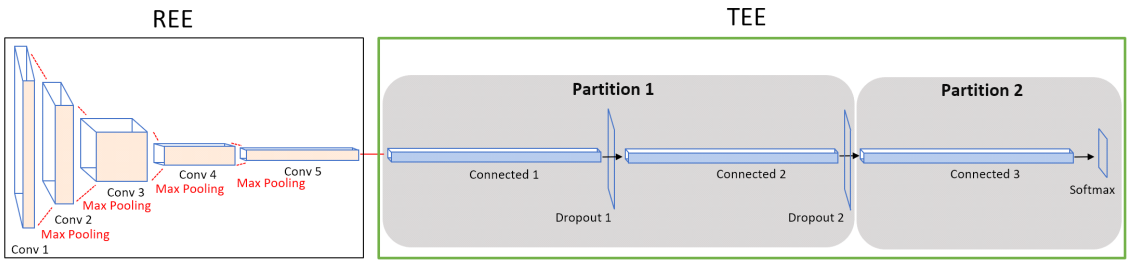


Fig. 10. Alexnet partitioned into 2 partitions with 4MiB secure memory for each partition.

는데 사용한 시간으로, 이는 user mode와 kernel mode로 나뉜다. TPMP와 DarkneTZ에서의 추론 중 소요되는 CPU 수행시간을 측정하기 위해 REE kernel mode time 및 REE user mode time을 측정한다. REE kernel mode time에는 TEE kernel time과 TEE user time을 포함한다. 따라서 Secure World인 TEE 내에서 계층이 수행되는 시간은 모두 Fig. 11의 kernel mode time에 나타난다. 그 결과로 나온 REE kernel mode time과 REE user mode time을 합한 것을 총 수행시간(run time)으로 칭한다. TEE 내 계층계수별(1개~6개)로 측정하였으며 계층 개수별 각 20번의 추론 수행시간의 평균을 산출한다.

DarkneTZ와 TPMP 모두 REE user mode time에서는 큰 차이를 나타내지 않는다. TEE내의 계층 개수가 많아질수록 REE user mode time은 소폭 감소한다. TEE내에서 수행되는 계층의 개수가 달라짐에따라, kernel mode time에서 큰 차이를 나타낸다. TEE내에서 수행되는 계층은 Fully Connected, Dropout 및 Softmax 계층인데, 이 중에서 Fully Connected 계층이 DNN 연산에서 대부분의 수행시간을 차지하는 계층이다. 또한 전체 가중치에서도 Fully Connected 계층의 가중치가 가장 큰 비중을 차지한다. 이러한 계층이 TEE내에서 연산되기 위해서는 해당 계층을 위한 가중치의 복호화와 계층 생성이 진행된다. Fully Connected 계층의 생성 시간은 계층당 평균 약 15.1초, 복호화 시간은 계층당 평균 약 0.9초로 측정되었다. 결론적으로 TEE내 Fully Connected 계층의 개수가 바뀜에 따라 kernel mode time 수행시간에 큰 차이를 보인다.

Fig. 11에서 DarkneTZ의 경우 TEE내 연산

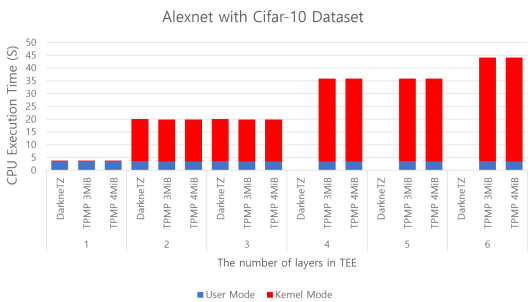


Fig. 11. CPU Execution time of Alexnet inference

되는 계층이 4개 이상인 경우 Secure memory 부족으로 인해 추론을 수행하지 못하므로 수행시간을 측정할 수 없다. TEE에서 연산 되는 계층이 3개 이하인 경우, 즉 DarknetTZ와 TPMP가 모두 DNN 연산 수행이 가능한 경우에는 DarkneTZ와 TPMP간의 수행시간 차이는 1% 미만으로, TPMP Manager를 통한 strategy 생성 과정이 추가되었음에도 매우 낮은 수준이다. 결론적으로 Fully connected 계층과 같은 많은 연산량을 가진 계층이 TEE내에 추가될수록 수행시간은 길어지나, 계층 partitioning을 통해 성공적으로 연산을 완료할 수 있다.

Fig. 9과 10에서 볼 수 있듯이, 6개의 계층을 TEE 내에서 수행하는 경우, 계층의 개수는 같지만, TEE secure memory의 크기에 따라 partitioning strategy(L[N])가 다른 경우가 존재한다. 이때, REE와 TEE간 world switching, 즉 SMC(Secure Monitor Call)이 한 번 더 발생하는데 . 이때 요구되는 수행시간은 평균 약 1ms로 전체 수행시간에 큰 영향을 주지 않는다. 따라서 partitioning strategy의(L[N]) 차이에 따른 전체 수행시간은 큰 차이를 보이지 않는다.

6.3 실험 결과 평가

DNN 연산을 수행하는데 요구되는 총 memory의 크기가 TEE secure memory보다 크더라도 해당 DNN을 partition을 하여 계층 단위로 수행이 가능하여, 기존 환경에서 수행이 불가능했던 큰 크기의 DNN 모델을 성공적으로 수행할 수 있다.

secure memory 크기가 사용자 환경 또는 하드웨어에 따라 달라지더라도 partitioning strategy를 유동적으로 조절한다. 따라서 한 개의 partition에서 수행되는 계층을 개수를 조건에 맞게 조절하여 성공적으로 연산을 수행한다.

결론적으로 TPMP는 DarkenTZ와 동일한 configuration에서 DNN 모델을 수행하는 경우, 더 많은 계층을 TEE내에서 수행하여 모델 정보와 같은 adversarial knowledge에 대한 공격 표면을 줄인다. 본 실험에서, TPMP는 Alexnet의 민감한 정보를 포함하는 Fully connected 계층을 모두 TEE내에서 수행하여 대부분의 민감한 파라미터를 보호할 수 있다. Fig. 12에서 볼 수 있듯이, TPMP는 6개의 계층을 TEE에서 수행할 때,

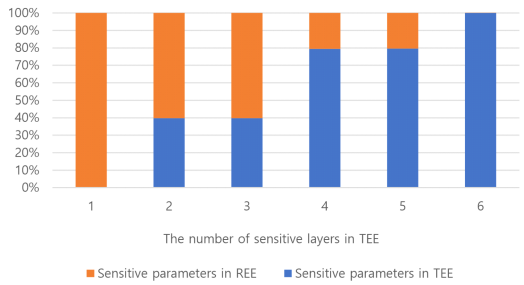


Fig. 12. Sensitive parameters safely protected in the TEE (Sensitive parameters in REE(6th) is too small to be shown)

Fully connected 계층에 대한 대부분의 파라미터를 보호할 수 있다.

DarkneTZ 대비, TPMP에서 발생하는 수행시간 증가는 1% 미만으로 무시할 만한 수준을 보여준다. 다만 TEE내에 크기가 큰 계층인 Fully Connected 계층의 포함 여부 및 그 개수에 따라 계층생성 시간과 가중치 복호화에 많은 수행시간이 추가로 소모된다. 이는 DarkneTZ에서도 동일하다.

VII. 관련 연구

TPMP 이전에도 딥러닝 연산의 기밀성을 보장하기 위한 다양한 연구들이 있었다. 먼저 동형 암호(homomorphic encryption)를 사용한 연구들이 있었다. 즉, 동형 암호는 별도의 복호화 과정 없이 암호화된 데이터에 대하여 연산을 수행할 수 있는 암호화 기법이기 때문에 동형 암호 기법으로 암호화된 데이터를 가지고 딥러닝 연산을 수행할 경우 딥러닝 모델에 대한 유출에 대한 우려 없이 딥러닝 연산을 수행할 수 있다. 하지만 이러한 동형 암호는 높은 계산 복잡성을 가지고 있어 상당한 성능 부하를 발생시키며[22][23] 특히 컴퓨팅 자원이 부족한 엣지 디바이스에서는 적용하기 어려운 방법이다.

이러한 한계로 인해 딥러닝 모델의 기밀성 보호를 위해 TEE를 이용하는 연구들도 있었다. 즉, TEE에서 딥러닝 연산을 수행하는 경우 외부의 REE에서 동작하는 코드들은 TEE 내부에 접근할 수 없으므로 기밀성이 보장되는 것이다. 예를 들어, Vessels[24]는 TPMP와 유사하게 DNN 모델의 계층별 종속성을 고려하여 TEE 내에서 수행되는 계층들의 memory 사용량을 최적화하였다. 하지만 Vessels은 TPMP와 다르게 TEE을 위해 ARM TrustZone이 아닌 Intel SGX[25]을 사용하였으며 일반적인 엣지 디바이스에서는 Intel SGX를 사용할 수 없다. 반면 DarkneTZ[11]은 TPMP와 마찬가지로 ARM TrustZone을 사용해 일부 민감한 DNN 계층들을 Secure World에서 수행되도록 하여 딥러닝 모델의 기밀성을 보호하였다. 하지만 DarkneTZ의 partitioning 기법은 DNN 계층들을 normal world와 secure world에서 수행되는 계층들로 이분법으로만 구분하였지만 TPMP은 Secure World에서 수행될 계층들도 memory 사용량에 따라 추가로 partitioning하여 결과적으로 TEE에서 더 많은 계층이 수행될 수 있도록 하였다. HybridTEE[26]와 PPFL[27]은 엣지 디바이스에서 마지막 계층 중 일부의 기밀성을 보호하기 위해 ARM TrustZone을 사용하고 나머지 계층들은 서버의 Intel SGX에서 수행되도록 하여 전체 딥러닝 모델의 기밀성을 보호하였다. 이때, 엣지 디바이스 측에서 수행되는 딥러닝 계층들의 기밀성 보장을 위해 TPMP를 사용할 경우보다 적은 secure world memory만으로 딥러닝 모델의 기밀성을 보장할 수 있을 것으로 예상한다.

DarkneTZ은 Train에 사용된 개인정보를 공격하는 MIA을 효과적으로 방어할 수 있도록 DNN 연산을 ARM TrustZone에서 수행하여 계층의 기밀성을 보장한다. 엣지 디바이스의 제한된 memory로 인해 가장 민감한 정보들이 담긴 마지막 계층만을 TEE에서 수행하여 공격자들이 계층의 매개변수에 접근할 수 없도록 한다. 하지만 전체 DNN의 대부분은 REE에서 수행되기 때문에 공격표면은 여전히 크다. 따라서 TPMP를 사용할 경우 더 많은 계층을 TEE에서 수행 가능하므로 공격표면을 크게 줄일 수 있다.

VIII. 고 찰

현재 TPMP는 계층 단위의 partitioning을 통해 TEE에서 수행되는 계층 전체의 크기가 TEE secure memory보다 크더라도 추론을 수행할 수 있음을 보여주었다. 다시 말해, 계층이 개수가 매우 많은 모델의 경우에도 해당 모델의 단일 계층의 크기가 TEE secure memory의 크기보다 작다는 조건에 부합하는 경우, 성공적으로 추론을 수행할 수 있다. 수행시간은 TEE에서 수행되는 계층 개수에 의한 계층 생성과 같은 절대적인 연산량, 가중치의 복

호화와 같은 동작에 영향을 받아 증가한다.

만약 단일 계층의 크기가 secure memory를 초과하는 크기를 가지는 모델의 경우 TEE 내에서 해당 계층생성이 불가능하여 추론을 수행할 수 없다. 이러한 경우, 모델 성능에 영향을 크게 미치지 않는 수준에서 노드 간 커넥션을 최소화하여 모델의 크기를 줄이는 프루닝(pruning) 또는 Sub-Layer partitioning[28]과 같이 하나의 계층을 노드 단위로 partitioning하여 연산을 수행하여 secure memory에 대한 dependency를 줄이는 등, 모델/뉴런 최적화 접근법들을 통해 정확도에 큰 영향을 미치지 않으면서도 DNN 모델을 TEE에서 수행 가능한 크기로 조절하여 수행하는 것에 관한 추가 연구가 필요하다.

IX. 결 론

본 논문은 ARM TrustZone을 활용하여 추론 과정에서의 DNN 모델의 기밀성을 효율적으로 보장해주는 TPMP를 제안하였다. 즉, TPMP는 동일한 설정 조건에서 기존 연구보다 더 많은 계층을 TEE 내에서 성공적으로 수행한다. 이는 DNN 모델을 이루고 있는 계층들의 memory 사용량을 토대로 계층 단위 partitioning을 수행하여 연산하기에 가능하다. 이러한 partitioning 결과를 토대로 제한된 memory상에서 다수 계층의 매개변수와 가중치의 기밀성이 보장될 수 있도록 하였다.

References

[1] Li E, Zhou Z, Chen X. "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy", pp. 31-36, Jun. 2018.

[2] Li H, Ota K, Dong M. "Learning IoT in edge: Deep learning for the internet of things with edge computing." IEEE network, vol. 32, no. 1, pp.96-101, Jan. 2018.

[3] Wu C, Brooks D, Chen K, et al. "Machine learning at facebook: Understanding inference at the edge", pp.331-344, Mar. 2019.

[4] Xu X, Ding Y, Hu SX, et al. "Scaling for edge inference of deep neural networks. Nature Electronics", Vol. 1. no. 4, pp.216-222, Apr. 2018.

[5] Sundararajan K, Woodard DL. "Deep learning for biometrics: A survey." ACM Computing Surveys (CSUR), vol. 51, no. 3, pp. 1-34, May. 2019.

[6] He K, Zhang X, Ren S, Sun J. "Deep residual learning for image recognition", pp.770-778, Dec, 2016

[7] Xiong W, Wu L, Alleva F, Droppo J, Huang X, Stolcke A. "The microsoft 2017 conversational speech recognition system", pp.5934-5938, Sep. 2018.

[8] Wang B, Gong NZ. "Stealing hyperparameters in machine learning", pp.36-52, July. 2018.

[9] Xie P, Ren X, Sun G. "Customizing trusted AI accelerators for efficient privacy-preserving machine learning". arXiv preprint arXiv:2011.06376. Nov. 2020.

[10] Molek, V., & Hurtik, P. (2020, August). "Training Neural Network Over Encrypted Data". In 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP) pp. 23-27, Sep. 2020.

[11] Mo F, Shamsabadi AS, Katevas K, et al. "Darknetz: Towards model privacy at the edge using trusted execution environments", pp.161-174, Jun, 2020

[12] Shokri R, Stronati M, Song C, Shmatikov V. "Membership inference attacks against machine learning models", pp. 3-18, Jun, 2017

[13] ARM Developer, "ARM Security Technology Building a Secure System using TrustZone Technology" https://developer.arm.com/documentation/PRD29-GENC-009492/c Apr. 2009.

[14] GitHub repository, "Fan Mo. DarkneTZ" https://github.com/mofanv/darknetz. Dec. 2020.

[15] Fredrikson M, Jha S, Ristenpart T. "Model inversion attacks that exploit confidence information and basic countermeasures", pp. 1322-1333, Oct. 2015.

[16] Biggio B, Corona I, Maiorca D, et al. "Evasion attacks against machine learning at test time", Springer, pp.387-402, Sep. 2013.

[17] Tramèr F, Zhang F, Juels A, Reiter MK, Ristenpart T. "Stealing machine learning models via prediction {APIs}", pp.601-618, Oct. 2016.

[18] Jun So-Hee, Lee Young-Han, Kim Hyun-Jun, Paek Yun-Heung. "A Study of AI model extraction attack and defense techniques". Proceedings of the Korea Information Processing Society Conference. 28(1), pp.382-384, May. 2021.

[19] cve.mitre.org. "Cve-2020-15205" https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-15205. Accessed .05.02, 2022.

[20] cve.mitre.org, "Cve-2021-37678" https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-37678. Accessed .05.02, sa2022.

[21] www.cs.toronto.edu "Cifar-10" https://www.cs.toronto.edu/~kriz/cifar.html. Accessed .01.25, 2022.

[22] Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J. "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy", pp.201-210, Feb. 2016.

[23] Hesamifard E, Takabi H, Ghasemi M. "Cryptodl: Deep neural networks over encrypted data". arXiv preprint arXiv:1711.05189. Nov. 2017.

[24] Kim K, Kim CH, Rhee JJ, et al. "Vessels: Efficient and scalable deep learning prediction on trusted processors", : pp.462-476, Oct. 2020.

[25] Costan V, Devadas S. "Intel SGX explained." Cryptology ePrint Archive, Jan. 2016.

[26] Gangal A, Ye M, Wei S. "HybridTEE: Secure mobile DNN execution using hybrid trusted execution environment", pp.1-6,Dec. 2020.

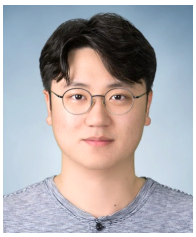
[27] Mo F, Haddadi H, Katevas K, Marin E, Perino D, Kourtellis N. "PPFL: Privacy-preserving federated learning with trusted execution environments", pp.94-108, Jun. 2021.

[28] VanNostrand PM, Kyriazis I, Cheng M, Guo T, Walls RJ. "Confidential deep learning: Executing proprietary models on untrusted devices." arXiv preprint arXiv:1908.10730. Aug. 2019.

〈저 자 소 개〉

송 수 현 (Suhyeon Song) 정회원
2021년 4월: Unitec Institute of Technology Computing System 학사 졸업
2022년 3월~현재: 부산대학교 융합보안대학원 석사
〈관심분야〉 정보보안, TrustZone

박 성 환 (Seonghwan Park) 학생회원
2021년 2월: 동서대학교 컴퓨터공학 학사 졸업
2021년 3월~현재: 부산대학교 융합보안대학원 석사
〈관심분야〉 시스템 보안, 정보 보안



권 동 현 (Donghyun Kwon) 정회원
2012년 2월: 서울대학교 전기컴퓨터공학과 학사 졸업
2019년 2월: 서울대학교 전기컴퓨터공학과 석박통합과정 졸업
2019년 3월~2020년 2월: 한국전자통신연구원 연구원
2020년 3월~현재: 부산대학교 정보컴퓨터공학부 조교수
〈관심분야〉 시스템 보안, 소프트웨어보안