연구논문/작품 중간보고서

2021 학년도 제 1학기

제목	클라우드 기반 머신러닝 서비스 보안 프레임 워크	○ 논문(○) 작품( ) ※해당란 체크
GitHub URL	https://github.com/Seo-han-gyeol/Graduate-Report.git	
평가등급	지도교수 수정보완 사항	팀원 명단
A,B,F중 택 1 (지도교수가 부여)	○ ○ ○	강 동 윤 (인) (학번: 2015312912) 서 한 결 (인) (학번: 2015311152)

2021 년 3월 19일

지도교수 : 이 호 준 서명

1. 요약

AI 기술이 적용된 서비스 제공에 강제되는 높은 메모리 사용량을 해결하기 위해 일반적으로 클라우드 컴퓨팅 기술을 이용합니다. 클라우드 기반 서비스는 애플리케이션 개발자로 하여금 메모리 사용량에 대한 걱정을 덜어주어 성능적인 부분을 좀 더 신경 쓸 수 있게 하며 이용자는 편리하게 양질의 서비스를 제공받을 수 있게 합니다.

하지만 보안 대책이 미흡한 클라우드 서비스는 서비스를 제공받아 얻는 이익만을 생각하기에는 보안사고로 인한 피해가 막대할 수 있습니다. AI 기술이 인간의 삶에 깊이 파고든 현 상황에서 우리가 사용하는 AI 기술이 적용된 애플리케이션 그 중에서 많은 부분을 차지하고 있는 클라우드 기반 애플리케이션의 보안은 그 중요도가 높다고 할 수 있습니다.

이를 위해 본 논문에서는 클라우드 기반 AI 서비스를 분석하여 어떤 공격이 이루어질 수 있는지 분석하고 그에 대한 연구된 방어법들의 효과를 확인하여 효과적인 것들을 선별하고 접목시키는 시도를 합니다.

2. 서론

2.1. 제안배경 및 필요성

인공지능(AI)에 대한 활발한 연구를 통해 비약적인 기술 발전이 이루어졌습니다. 자율주행 자동차, 스마트폰의 안면 인식, 여러 빅데이터를 처리하는 것 등에 이미 AI 기술이 적용되고 있습니다. 사람이 구별하기 힘든 이미지를 인식하고, 방대한 양의 데이터에서 쓸모 있는 정보를 추출하는 등 인간의 능력을 넘어서는 기술들이 등장하며 기업에서는 모바일 또는 IoT 애플리케이션에 이런 AI 기술을 접목시켜 다양한 서비스를 제공하려는 노력을 하고 있습니다.

몇몇 기업들은 모델을 스스로 train 시킬 능력이 부족한 사용자에게 기업의 모델을 서비스로 제공해주곤 하는데, 이 때 모델의 parameter 등의 모델에 대한 정보는 사용자로부터 숨겨져 있지만, 사용자는 해당 모델을 이용하여 input에 대한 output을 얻을 수 있고, 이 결과를 이용해 모델의 기밀성이 낮아질 수 있습니다. 모델의 기밀성이 낮아지면 모델을 copy 하여 비슷한 성능을 가진 모델을 만들어 기업의 경제활동에 큰 피해를 안길 수 있고, 또한 copy 된 모델을 이용하여 연구함으로써 원래의 모델에 악의적으로 잘못된 input을 집어넣

어 원래의 모델의 성능을 크게 떨어뜨릴 수도 있습니다. 그렇기 때문에 모델의 parameter나 모델의 종류 등의 모델의 정보의 기밀성을 유지하는 것은 중요합니다.

또한, 기업에서는 빠른 계산능력과 높은 정확도를 가진 모델을 사용하여 양질의 서비스를 제공하고자 하지만, 이는 많은 메모리 사용이 불가피합니다. 보통 클라우드 컴퓨팅 기술로 이를 해결하려 하지만 보안 대책이 미흡한 클라우드 서비스는 여러 보안 문제가 발생할 수 있습니다. 앞으로 AI 기술은 더욱 발전하여 우리 생활의 모든 방면에서 사용될 것이 분명한 이 시점에서 보안사고로 인해 제공한 개인 정보가 보호되지 못할 수도 있다는 것은 굉장히 심각한 문제입니다.

클라우드 기반 머신 러닝 서비스에 대한 공격자는 크게 외부 공격자와 내부 공격자로 분류할 수 있습니다.

외부 공격자란 모델에 대한 정보(모델의 종류, parameter 등)을 전혀 모르고 입력에 대한 결과만 알고 있는 상태에서 이루어지는 black-box attack을 시행하는 사람을 말합니다. 대부분의 공격은 black-box 환경에서 이루어지며, 그만큼 black-box attack은 굉장히 많은 곳에서 사용될 수 있습니다. 예를 들어, 모델을 공격하여 잘못된 판단을 하도록 유도할 수 있는데, 자율주행 차량을 속여 빨간색 신호등을 초록색 신호등으로 판단하도록 유도하여 사고를 유발하거나, 컴퓨터 등의 기계에 설치된 백신을 침투된 바이러스를 정상적인 프로그램으로 인식하도록 유도하여 컴퓨터를 망가뜨리는 등 무궁무진한 방법으로 사용될 수 있습니다. 따라서, black-box attack을 사용하는 외부 공격자에 대한 방어대책을 강구하는 것이 중요합니다.

반면 내부 공격자는 모델에 대한 정보를 알고 있는 상태에서 이루어지는 white-box attack을 시행하는 사람을 말하며 나쁜 마음을 먹은 서버 관리자를 예로 들 수 있습니다. white-box attack은 모델의 parameter와 같은 모델에 대한 모든 정보를 알고 있는 상태로 시작하기 때문에, 모델의 알고리즘, 학습된 data의 분포 등을 이용하여 모델의 취약점을 쉽게 알 수 있고, 이로 인해 쉽게 공격을 수행할 수 있기 때문에 black-box attack보다 사용할 수 있는 곳은 한정되어 있지만 훨씬 치명적인 공격을 수행할 수 있습니다. 따라서 white-box attack을 사용하는 내부 공격자에 대한 방어대책 역시 강구해야 합니다.

2.2. 연구논문의 목표

본 논문의 목표는 크게 세 가지입니다.

첫째, 머신 러닝에 관한 대략적인 특성과 실행되는 알고리즘 등에 대해서 관련 논문을 찾아보고 파악합니다. 이에 대해 알아야 모델에 관한 취약점이 무엇이 있는지 알 수 있고, 이 취약점이 어떤 이유로 인해 모델을 공격할 수 있는지에 대해 알 수 있기 때문에 이 과정은 본 논문에서 꼭 다뤄져야 합니다.

둘째, AI 모델이 갖는 여러 취약점을 다양한 논문으로부터 분석하고, 그 취약점을 사용하여 실제로 공격이 가능한 지 실험을 합니다. 만약 해당 취약점이 실험을 해보았을 때 공격이 잘 되지 않는다면, 왜 제대로 되지 않는지 그 이유를 분석하고 보완합니다. 만약 해당 취약점이 실제로 공격을 잘 수행한다면, 해당 취약점을 방어할 수 있는 방어법을 여러 논문으로부터 발표되거나 해당 취약점에 대한 알고리즘을 분석해 어느 부분이 취약했는지 알아내는 등의 방법으로 찾아내 그 방어법을 직접 적용해 봄으로서 그 효과를 확인합니다.

마지막으로, 앞선 취약점과 방어법 중 직접 실험하여 실제로 공격이 가능하고, 또한 그 공격에 대해 실제로 효과적으로 방어가 가능한 여러 효과적인 방어법들을 접목시켜 여러 취약점들을 방어할 수 있는 하나의 프레임워크를 제시합니다. 그리고 해당 프레임워크를 실제로 사용했을 때 그 메모리 사용량이나 속도 등 그 실용성을 실험하고 실제로 적용할 수 있는 프레임워크인지 판단합니다.

2.3. 연구논문 전체 overview

여러 논문으로부터 찾아낸 취약점들을 설명하고, 또 해당 취약점을 막을 수 있는 방어법들을 다양한 논문으로부터 찾아내어 나열합니다. 이렇게 찾아낸 여러 취약점들과 방어법들을 실제로 모델에 접목시켜 실험을 한 후, 실제로 공격이 수행 가능한 취약점을 찾고, 해당 취약점을 효과적으로 방어할 수 있는 방어법들을 한데 모아 접목시킨 하나의 프레임워크를 제시하는 것이 본 논문의 전체적인 overview입니다. 저희가 여러 논문으로부터 찾아낸 취약점과 그에 해당하는 방어법은 크게 3가지가 있으며, 그 종류는 아래의 표와 같습니다.

[표 1] AI 모델의 취약점과 방어법

취약점	방어법
surrogate data를 이용한 model hyperparameter stealing attack	surrogate data가 입력될 때 mislabel
adversarial example을 이용 (인간이 인지하지 못하는 noise를 추가)	adversarial example을 학습데이터로 활용
클라우드 서버 관리자에 의한 private data를 탈취하는 공격	Enclave를 만들어 TEE 환경 구축

첫 번째 취약점으로는 surrogate data를 이용한 model hyperparameter stealing attack이 있습니다 [1]. 이 취약점은 예를 들면 강아지의 종을 분류하는 모델이 있다면 그 모델에 강아지가 아닌 다른 동물, 이를테면 고양이의 이미지를 input으로 넣고, output으로 나온 결과를 이용해 model copy를 가능케 하는 취약점입니다.

이에 대한 방어법으로는 model copy를 위한 surrogate data가 입력되었을 시 일부러 mislabel 시키는 방법이 있습니다 [2]. 간단하게 말해서, 예를 들어 앞서 말한 강아지의 종을 분류하는 모델에 강아지가 아닌 고양이의 이미지가 input으로 들어온다면, 이를 눈치채고 일부러 잘못된 output을 출력하게 함으로써 model copy를 제대로 수행하지 못하게 하는 것입니다.

두 번째 취약점으로는 adversarial example을 이용한 공격, 즉 인간이 인지하지 못하는 수준의 noise를 input image에 추가하여 mislabel 시키는 공격이 있습니다. 원래의 image와 인간이 인지하지 못하는 수준의 noise가 추가된 image는 인간의 눈으로 보기엔 거의 같은 사진이므로, 통상적으로 모델에 input으로 집어넣을 경우 같은 output이 출력되어야 합니다. 하지만 모델은 이 차이를 인식하여 다른 output이 출력되므로, 모델이 잘못된 결과를 내도록 유도하여 성능의 저하를 일으킬 수 있습니다.

이에 대한 방어법으로는 adversarial example을 학습데이터로 활용해서 공격자가 adversarial example을 이용해 공격했을 때 mislabel될 확률을 줄이는 방법이 있습니다. 간단히 말해서, 인간이 인지하지 못하는 수준의 noise가 추가된 image를 모델을 만들 때부터 input으로 집어넣어 모델이 이러한 adversarial example에 대한 대책을 가지게 하는 것입니다.

세 번째 취약점으로는 클라우드 서버 관리자에 의한 private data를 탈취하는 공격이 있습니다. AI 모델은 많은 메모리 사용량 때문에 클라우드 플랫폼을 많이 사용하는데, 클라우드를 이용하는 서비스 특성 상 여러 보안 관련 문제가 존재합니다. 여기서 만약 클라우드 서버 관리자가 악의적으로 사용자의 input을 도용하는 등 AI 모델의 무결성을 침해할 수 있는 문제가 있습니다.

이에 대한 방어법으로 Intel SGX와 같이 공격으로부터 엄격하게 보호되는 Enclave 공간을 만들어 클라우드 서버 관리자도 Enclave 내부의 민감한 데이터에는 접근하지 못하게 하는 방법이 있습니다 [5]. 하지만 Enclave 공간을 만들어 AI 모델을 실행시키는 것은 안전하지만 그만큼 심각한 속도 저하가 발생하게 됩니다. 이를 해결하기 위해, 모델의 parameter를 Enclave 외부에 두고 On-demand 형식으로 parameter를 load하고, 모델 inference 과정에서 발생하는 Convolution layer의 computation의 느린 연산 속도를 해결하기 위해 portioned convolution을 도입하는 등 여러가지 시도가 존재합니다 [5]. 하지만 해당 논문에서 제안한 프레임워크에서는 모델의 parameter를 Enclave 외부에 두기 때문에 parameter는 제대로 보호할 수 없어 model copy가 가능하다는 문제가 있고, 따라서 앞서 언급된 여러 취약점들과 그에 대응되는 방어법들을 다같이 적용하여 이 문제를 해결하는 것을 기대할 수 있습니다.

여러 논문들을 통해 여러가지 취약점을 찾았고, 관련 논문에서 연구된 여러가지 방어법들을 찾았습니다. 본 논문에서는 이들을 접목시켜 새로운 프레임워크를 만드는 시도를 했고 결과적으로 앞서 언급된 취약점들을 효과적으로 방어할 수 있다는 사실을 실험을 통해 확인하였습니다.

2.4. 요약

AI 기술에 대한 활발한 연구를 통해 여러 분야에서 AI 기술들이 사용되고 있고, 몇몇 기업들은 모델을 스스로 train 시키기 힘든 사용자에게 기업의 모델을 서비스하는데, 이 과정에서 사용자가 모델의 여러가지 방법으로 model의 parameter를 탈취해 모델을 copy할 수 있

습니다. 또한, 한정된 메모리 용량 때문에 클라우드 상에서 모델을 실행하는 경우가 많은데, 이 과정에서 서버 관리자에 의해 private data가 탈취되는 등의 여러 보안 상의 문제가 존재합니다.

이러한 여러가지 문제 때문에 저희는 본 논문을 통해 여러 취약점을 찾고 이를 방어할 수 있는 방어법들을 통합하여 접목시킨 하나의 프레임워크를 제시하고, 실제로 앞서 언급된 취약점들을 효과적으로 방어할 수 있는지 실험을 합니다.

3. 관련 연구

적대적 공격(Adversary attack)이란 AI 모델에 의도적으로 적대적 교란(Adversarial Pertubation)을 일으켜 잘못된 결과를 내놓도록 하는 공격을 의미합니다 [4]. 이러한 적대적 공격의 종류에는 회피 공격(evasion attack), 중독 공격(poisoning attack), 탐색적 공격(exploratory attack)이 있습니다 [1].

회피 공격은 공격에 의해 변화된 output을 사람이 인지할 수 있는 지의 여부에 따라 2가지로 나뉩니다. 인지하지 쉬운 공격은 원래의 input에 특정 패턴을 나타내는 스티커를 결합해 새로운 input을 만드는데, 이 경우 인간이 input이 변화되었다는 사실을 인지할 수 있고 두 input이 모두 같은 output을 도출해야 한다는 것 역시 알 수 있지만, 기계는 두 input에 대해 다른 output을 도출합니다. 인지하기 어려운 공격은 원래의 input에 미세한 노이즈를 섞어 인간은 무엇이 변화되었는지 인지할 수 없지만, 기계는 이 노이즈를 인지하고 서로 다른 output을 도출합니다.

중독 공격은 공격자가 AI 모델 자체에 침투하여 모델의 성능 저하를 일으키는 공격입니다. 대표적으로 모델의 dataset을 손상시켜 mislabel 시키는 공격이 있습니다. 이 공격의 예로는 스팸 메시지를 걸러내는 필터를 손상시키거나, 생체 인식 기계를 손상시켜 잘못된 결과를 도출하게 만드는 공격 등이 있습니다.

마지막으로, 탐색적 공격은 크게 model inversion attack과 model extraction attack으로 구분할 수 있습니다 [1]. model inversion attack은 모델의 train에 사용된 data를 탈취하는 공격입니다 [8]. 만약 이 데이터가 기업의 기밀 정보 등의 민감한 데이터라면 많은 피해가 발생하게 됩니다. model extraction attack은 AI 모델의 이름, parameter 등 모델의 정보를 탈취하는 공격입니다. 모델의 parameter를 탈취하면 모델의 알고리즘이 공개되는 등 기밀성이 저하되어 후에 회피 공격이나 model inversion attack을 수행할 수 있게 합니다. 이 공격은 원래 모

델의 정보에 대해서 모르지만 다른 방법으로 이 정보를 탈취하는 공격이기 때문에 black-box attack에 해당됩니다.

모델의 취약점을 공격할 때 공격자가 가지고 있는 모델에 대한 정보의 보유량에 따라 white-box attack과 black-box attack으로 나뉩니다. 하지만 white-box attack 같은 경우는 input data와 그에 따른 output을 제외한 모델의 다른 정보들 역시 다 알고 있는 상태를 가정하므로, 실제로 대부분의 공격은 black-box attack으로 이루어집니다. black-box attack은 크게 3가지 종류로 나눌 수 있는데, 각각 Non-Adaptive(비적응형) black-box attack, Adaptive(적응형) black-box attack, Strict(엄격한) black-box attack으로 구분할 수 있습니다.

Non-Adaptive black-box attack 이란 모델의 학습 데이터 분포(Training Data Distribution)에만 접근이 가능한 공격입니다. 해당 학습 데이터 분포를 이용해 표본을 추출하고, 이를 통해 dataset을 생성합니다. 이 dataset을 활용해 새로운 모델(Local Model)을 훈련시킵니다. 이 모델을 이용해 adversarial example을 생성하고 이를 원래의 모델에 적용하여 잘못된 분류를 하도록 유도합니다.

Adaptive black-box attack 이란 공격자가 모델에 접근한 공격입니다. 모델에 input을 넣고 그에 따른 output을 확인할 수 있습니다. 이렇게 생성된 input과 output의 쌍을 이용해 Non-Adaptive black-box attack과 같이 새로운 모델을 만들고 Adversarial Example을 생성하여 원래의 모델에 오분류를 일으킵니다.

Strict black-box attack 이란 Adaptive black-box attack과 같이 input과 output의 쌍을 알 수 있지만, input의 값을 임의로 변경시켜 다른 input에 대한 출력값을 알 수 없습니다.

AI 모델에는 여러 가지 취약점이 존재합니다. 첫 번째로, surrogate data를 이용한 model hyperparameter stealing attack 이 있습니다 [1]. model의 parameter를 도용하게 되면 이후에 학습자의 지적 재산권과 알고리즘의 기밀성이 저하되고, 공격자가 evasion attack 또는 model inversion attack 등의 공격을 수행할 수 있게 됩니다. 여기서 surrogate data 라는 것은, 예를 들어 강아지의 종을 분류하는 모델을 만들어 여러 강아지의 사진을 input data로 넣어 model의 hyperparameter를 구했을 때, 공격을 위해 강아지가 아닌 고양이의 사진을 input data로 넣는데, 이 때 고양이의 사진이 surrogate data 가 됩니다. 이 때 고양이의 사진을 input으로 넣어 구한 모델과 원래의 모델인 강아지의 사진을 input으로 넣은 모델을 비교해서 만약 test data를 넣었을 때 두 모델의 결과가 비슷하다면 성공적으로 hyperparameter을

도용해서 model copy에 성공했다는 것을 알 수 있습니다.

이에 대한 방어법 역시 존재하는데, surrogate data 가 input data로 주어졌을 때 일부러 mislabel 시킴으로써 model copy를 제대로 실행하지 못하게 하는 방법이 있습니다 [2]. surrogate data의 여부를 확인하는 방법은 존재하는 공격들이 대부분 Out-of-Distribution (OOD) query를 생성한다는 사실을 이용합니다. OOD로 간주되는 query에 대해 mislabel을 시키면, 공격자의 dataset (surrogate data)의 상당수에 mislabel이 되어 이 data를 이용해 train을 시킬 경우 정확도가 상당히 낮은 결과가 발생하게 됩니다. 때문에 surrogate data를 이용하여 model copy를 하는 것을 방어할 수 있게 됩니다.

또한 널리 알려진 취약점으로 Fast Gradient Sign Method(FGSN) 이 있는데, input data에 인간이 인지하지 못하는 수준의 noise를 추가해 mislabel을 시켜 train이 정상적으로 되지 않게 하는 방법이 있습니다 [3]. 이 취약점의 원리는 model이 사물을 구별하는 decision boundary 근처의 값을 조금씩 바꾸면 컴퓨터는 이 차이를 쉽게 구별할 수 있지만 인간의 기준으로는 이 차이를 구별하기 힘들다는 것입니다. 따라서 인간은 차이가 없는 정상적인 input data로 인식하지만 기계는 mislabel 되어 낮은 정확도의 model이 만들어질 수 있습니다.

이에 대한 방어법으로 Defense-GAN이라는 기법이 있습니다 [6]. 먼저, GAN(Generative Adversarial Network)이란 딥러닝 알고리즘의 하나로서, 생성자(Generator)와 식별자(Discriminator)가 서로 대립하여 결과적으로 모델의 성능을 향상시키는 기법입니다. 생성자는 데이터를 받아 해당 데이터와 비슷한 가짜 데이터를 생성하고, 식별자는 원본과 가짜 데이터를 식별하는 역할을 합니다. 서로를 경쟁시킬수록 생성자는 원본과 굉장히 비슷한 가짜 데이터를 만들게 되고, 식별자는 그 가짜 데이터도 식별할 수 있는 능력을 갖추게 됩니다. Defense-GAN은 이런 GAN 알고리즘을 사용한 기법으로, 원래 이미지에 noise가 추가된 adversarial example을 학습 데이터로 활용합니다. defense-GAN은 원래 이미지와 adversarial example의 차이를 최소화하는 새로운 이미지를 생성하고, 이 이미지를 이용해 GAN 알고리즘을 수행합니다. 결과적으로 원래의 이미지에 noise가 추가된 adversarial example가 정상적인 이미지로 인식되어 noise가 사라지게 되어 adversarial example을 이용한 공격을 방어할 수 있습니다.

클라우드 서버 관리자에 의한 private data를 탈취하는 공격도 존재합니다. 몇몇 기업에서는 부족한 메모리 용량 때문에 클라우드 서비스를 이용하여 소비자에게 AI 모델을 제공하는

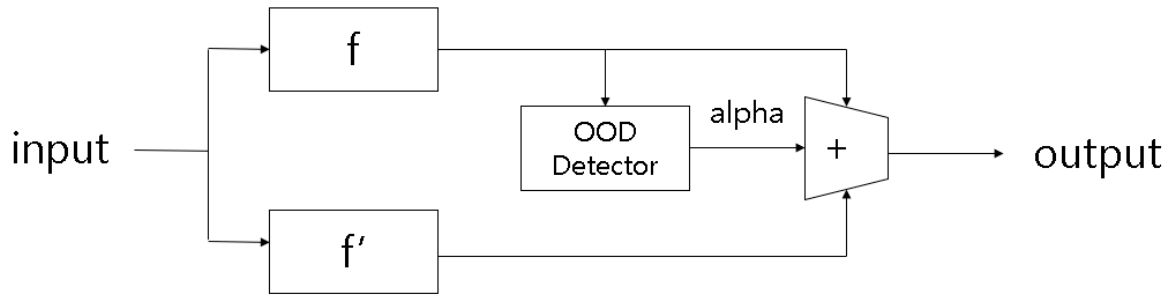
데, 이 과정에서 사용자의 private data가 노출이 되고, 클라우드 서버 관리자가 악의적으로 이 데이터를 탈취할 수 있습니다. 또한, 클라우드의 용량 역시 무한하지 않고 속도도 느려진다는 단점이 있습니다.

이에 대한 방어법으로 Intel SGX, ARM trustzone, AMD 등을 이용하여 신뢰 실행 환경(Trusted Execution Environment, TEE)를 구축하여 클라우드 상에서의 private data 유출을 막는 방법이 있습니다. 여기서 TEE란 신뢰할 수 있는 격리된 환경을 이용하여 프로그램의 기밀성과 무결성을 제공하는 기술입니다 [7]. 따라서 TEE를 이용해 클라우드 내에 일반 영역과 보안 영역 두 영역으로 나눈 뒤 보안 영역에 사용자의 private data를 저장하고 모델을 train한다면 클라우드 서버 관리자가 private data를 탈취할 수 없게 됩니다.

4. 제안 작품 소개

본 논문의 최종 목표인 클라우드 기반 AI 서비스 보안 프레임워크를 위해 조사한 AI 모델의 3가지 취약점 중 Adversarial example을 이용한 공격은 그 방어법이 model train 과정에서 이루어 지므로 서비스될 App의 보안 프레임워크에는 나머지 두 가지 취약점에 대한 방어법을 적용시킵니다.

4.1. 첫 번째 취약점인 surrogate data를 이용한 model stealing attack에 대한 방어법으로 참고논문 [2]에서 제시한 Adaptively Injecting Misinformation을 자세히 소개하겠습니다.



[그림 1] Adaptive Misinformation 알고리즘

AM(Adaptive Misinformation)은 [그림 1]과 같은 구조를 띄고 있습니다. 가장 처음 input x에 대해서 OOD(Out Of Distribution) Detector가 해당 input이 ID(In Distribution)인지 OOD인지 원래 모델 f의 결과값 f(x)를 보고 판단합니다. 만약 ID라면 원래 모델 f에 의한 결과값을 output으로, OOD라면 mislabel 시키는 모델 f'에 의한 결과값을 output으로 줍니다.

4.1.1. OOD Detector

OOD Detector는 AM 알고리즘 핵심 아이디어입니다. f(x) 값을 Soft Max Probability로 나타냈을 때, 그 중 예측값이 될 최댓값이 특정값(임계치) 이상인지 아닌지를 확인함으로써 input x가 OOD인지 ID인지 판단합니다. Soft Max Probability의 최댓값은 해당 값의 index가 예측값이 될 확률로 볼 수도 있는데, 그 확률이 낮을수록 input이 OOD일 확률은 높아진다는 사실로부터 나온 아이디어입니다. 판단 결과를 값 alpha로 return 하는데, 이 alpha값은 4.1.3. output에서 설명하겠습니다.

4.1.2. f'

f'은 OOD input에 대해 mislabel 시키기 위한 모델로써 보통 정답을 맞출 확률을 높이는 train에서 사용하는 cross entropy loss function [식 1] 과 달리 [식 2]를 사용하여 정답을 맞출 확률을 낮추는 train을 시킵니다.

$$loss = -\sum_i t_i log(s_i)$$

[식 1]

$$loss = -\sum_i t_i log(1 - s_i)$$

[식 2]

* t_i는 i번째 answer, s_i는 예측값을 soft max probability로 나타낸 것의 i번째 element

[식 2]에서 s_i는 soft max probability 이므로 [0,1] 범위의 값을 가집니다. train 과정에서 1에 가까울수록 해당 index가 답이 될 확률이 높다고 판단하는데, 1 - s_i의 경우는 정확히 그 반대의 의미를 가지게 됩니다. 즉, 정답이 될 확률이 가장 높은 index의 값을 가장 낮게 만들어 해당 index가 정답이 될 확률을 낮출 수 있게 됩니다.

4.1.3. output

마지막 output은 OOD detector에서 판단한 결과를 바탕으로 ID라면 f(x), OOD라면 f′(x)로 결정되어야 합니다. 이를 반영한 식은 [식 3]과 같습니다.

$$output = (1 - \alpha) * f(x) + \alpha * f'(x)$$

$$\alpha = S(ymax - \tau)$$

$$S(z) = \frac{1}{1 + e^{\nu z}}$$

[식 3]

S(z)는 reverse sigmoid function으로 z값이 0보다 크면 0.5보다 작은 값을 반환하고 0보다 작으면 0.5보다 큰 값을 반환합니다. 따라서 OOD Detector에서 ymax - τ(임계치) 값을 확인하여 결정된 α값으로 인해 ID일 경우 (ymax - τ > 0 일때, α -> 0 이면 output -> f(x)) f(x) 값이, OOD일 경우 (ymax - τ < 0 일때, α -> 1 이면 output -> f′(x)) f′(x)값이 output이 됩니다.

4.2. Intel SGX를 이용한 private data 보안

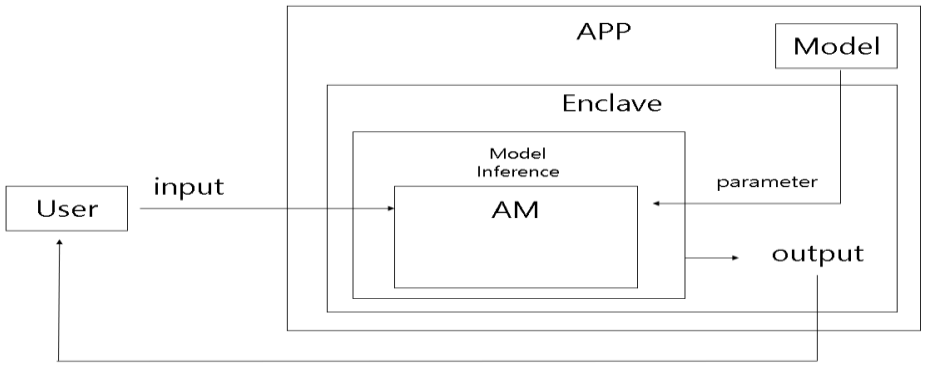
Intel SGX(Software Guard Extensions)는 Intel CPU에 적용된 코드 및 데이터를 메모리내에 격리하는 하드웨어 기반 보안 기술입니다. Enclave라고 하는 private memory를 할당하여 해당 메모리 사용자를 제외한 다른 그 어떤 누구도 접근할 수 없도록 설계되어 있습니다. 서비스를 이용하는 고객들의 개인정보는 이를 훔치려는 해커뿐만 아니라 서버 관리자로부터 또한 보호되어야 합니다. private data 보안을 위해 참고 논문 [5]에서는 Intel SGX를 이용한 방어법 Occlumency를 소개하고 있습니다.

Occlumency는 Intel SGX를 이용해 TEE 환경을 구축하고 공격으로부터 강하게 보호되는 Enclave 공간을 할당하여 그 안에서 model inference를 실행하는 방식입니다. 참고 논문 [5]에서는 모델 추론 과정 전체를 Enclave 내부에서 실행하여 클라우드 서버 관리자에 의한 공격을 방어합니다. 하지만 enclave를 만들 수 있는 공간의 크기는 크지 않기 때문에 model의 parameter를 보호하지 않기로 결정하고 이를 enclave 외부에 두어 enclave 공간의 사용량을

줄였습니다. 그 후 필요한 parameter만 그 때 즉시 on-demand 형식으로 enclave에 load 하는 방법을 사용하였습니다. 하지만 모델의 parameter를 보호하지 않기 때문에 모델이 오염될 수 있고, 이로 인해 오염된 parameter가 load 되는 문제가 발생할 수 있습니다. 이를 방지하기 위해 해당 논문에서는 Hash-checking을 도입하여 모델의 무결성을 확인합니다. 속도가 느려진다는 단점을 해결하기 위해 parallel pipeline과 partitioned convolution을 도입하였는데, parallel pipeline은 convolution 과정에서 발생하는 bottleneck(병목) 현상을 억제하기 위한 방법입니다. 머신러닝의 과정은 크게 parameter loading, hash-checking, model inference 세 가지로 구분되는데, 만약 한 parameter가 온전히 저 과정을 끝낸 후에 다음 parameter를 load한다면 너무나 많은 시간이 걸리기 때문에, ring buffer를 이용하여 parallel pipeline을 구축하여 parameter를 load하는 과정을 단축시킵니다. 머신러닝 과정에서 여러 번의 행렬 convolution 연산을 실행하는데, 보통 이 연산은 크기가 너무 커서 enclave의 용량을 초과합니다. 이를 해결하기 위해 행렬을 partition하여 연산을 하는데, 이 과정을 거치면 그만큼 할당되는 메모리의 용량이 감소합니다. 예를 들어, 2개로 partition 하면 2배로 적은 메모리가 필요하고, 4, 8, ... 개로 partition하면 4, 8, ... 배로 적은 메모리가 필요하게 됩니다.

4.3. Framework

목표하는 framework 구성은 [그림 2]와 같습니다. 서비스를 제공할 모델을 train 시킬 때 GAN 알고리즘을 적용하여 adversarial example attack을 대비하고, App실행 과정에서 AM과 model inference 전부 enclave 내부에서 진행하여 model stealing attack과 private data steal을 모두 방어합니다.



[그림 2] Framework

5. 구현 및 결과 분석

5.1. AM 실험 모델

AM을 실행해볼 모델로 사용하기 위해 kaggle dog-breed-identification (https://www.kaggle.com/c/dog-breed-identification/data)에서 데이터를 가져와 추려서 6898개의 강아지 이미지로 80개의 품종으로 분류하는 classifier model을 train 시켰습니다. Pytorch를 이용한 코드를 통해 전체 6898개의 이미지 중 5%인 345개의 validation set을 대상으로 accuracy 82.6%인 model을 만들어낼 수 있었습니다.

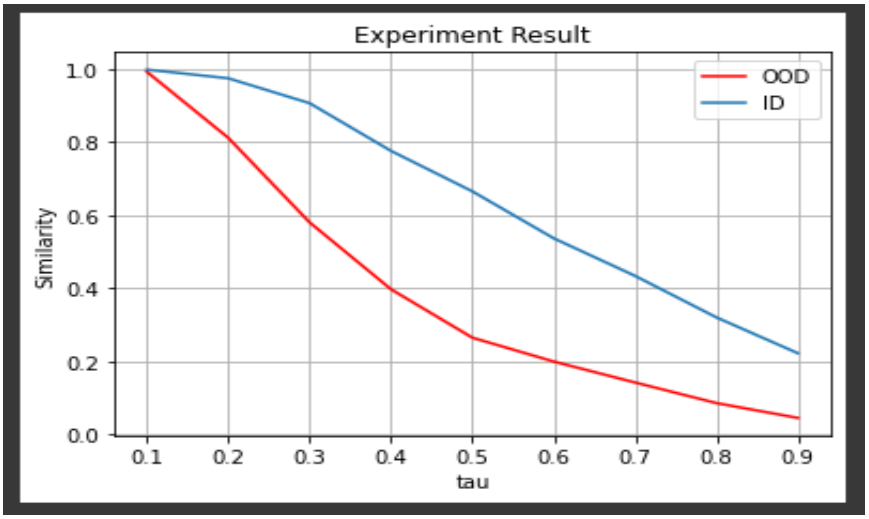
original model에서는 loss function을 torch.nn.CrossEntropy를 바로 사용하였지만, mislabel model을 구현할 때는 CrossEntropy가 LogSoftmax와 NLLLoss의 combine임을 고려하여 loss function을 x=torch.nn.Softmax, torch.log(1-x), torch.nn.NLLLoss의 순차적방식으로 구현했습니다.

```
train_loss: 0.0231, val_loss: 0.7242, val_acc: 0.8261
Finish Training.
```

[그림 3] original model의 train

```
train_loss: 0.0019, val_loss: 0.0000, val_acc: 0.0000
```

[그림 4] mislabel model의 train



[그림 5] 유사도 그래프

[그림 5] 임계치 τ의 값에따라 OOD input과 ID input을 넣었을 때, AM을 적용했을 경우와 하지 않았을 경우 각각의 유사도 그래프. test image는 kaggle cat and dog (https://www.kaggle.com/tongpython/cat-and-dog)을 사용

5.2. 분석

[그림 5]의 그래프를 보면 τ값이 작을 수록 OOD input을 판단하는 능력이 떨어지지만, τ값이 커질수록 ID input을 판단하는 능력도 떨어지게되는 것을 확인할 수 있습니다. 따라서 적절한 τ값의 선택이 AM 알고리즘의 성능을 높이는데 가장 중요한 요소라고 할 수 있습니다. 해당 그래프에서는 τ=0.3 이 ID input에 대해서는 유사도 90.60%, OOD input에 대해서는 유사도 57.96%로, 원래 모델의 성능을 크게 저하시키지 않으면서 attacker의 공격 능력은 크게 저하시키는 가장 적절한 임계치라고 분석됩니다.

6. 결론 및 소감

현재 인공지능은 굉장히 많은 분야에서 사용되고 있습니다. 더구나 부족한 메모리 용량 때문에 여러 기업에서는 클라우드를 이용해 사용자에게 모델을 서비스하는데, 이 과정에서 여러 취약점들이 발견되었습니다. 본 논문에서는 해당 취약점들을 분석하여 그에 따른 방어법들을 적용한 보안 프레임워크를 만드는 것이 목표였고, 여러 논문에서 연구된 방어법을 적용했을 때 실제로 취약점을 효과적으로 방어할 수 있는 것을 확인할 수 있었습니다. 학부수업에서 여러 인공지능 관련 과목들을 수강할 때는 알지 못했는데, 본 논문을 쓰며 여러 취약점들을 조사하며 인공지능 분야에서도 역시 보안이 중요하다는 사실을 깨달을 수 있었습니다.

7. 참고 문헌

[1] B. Wang and N. Z. Gong, "Stealing Hyperparameters in Machine Learning," in IEEE 2018.

[2] S. Kariyappa and M. K. Qureshi, "Defending Against Model Stealing Attacks with Adaptive Misinformation," in CVPR 2020.

[3] 이슬기, 김경한, 김병익, 박순태, "기계학습 모델 공격연구 동향: 심층신경망을 중심으로," 정보보호학회지 2019.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples", in ICLR 2015.

[5] T. Lee, Z. Lin, S. Pushp, C. Li, Y. Liu, Y. Lee, F. Xu, C. Xu, L. Zhang, J. Song, "Occlumency: Privacy-preserving Remote Deep-learning Inference Using SGX", in ACM ISBN 2019.

[6] P. Samangouei, M. Kabkab, R. Chellappa, "Defense-GAN: protecting classifiers against adversarial attacks using generative models", in ICLR 2018.

[7] 전상기, 최창준, 이종혁 (2017). Trusted Execution Environment의 구성 요소 및 응용 기술 조사. 한국통신학회 학술대회논문집, 65-66

[8] 류권상, 최대선, "인공지능 보안 공격 및 대응 방안 연구 동향", 정보보호학회지 2020.