*Journal of The Korea Institute of Information Security & Cryptology* ISSN 1598-3986(Print)
VOL.29, NO.1, Feb. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.1.117

# 공개 딥러닝 라이브러리에 대한 보안 취약성 검증*

정 재 한,[1†] 손 태 식[1,2‡]
[1]아주대학교 컴퓨터공학과, [2]아주대학교 사이버보안학과

# Security Vulnerability Verification for Open Deep Learning Libraries*

JaeHan Jeong,[1†] Taeshik Shon[1,2‡]
[1]Department of Computer Engineering, Ajou University
[2]Department of Cyber Security, Ajou University

요 약

최근 다양한 분야에서 활용중인 딥러닝은 적대적 공격 가능성의 발견으로 위험성이 제기되고 있다. 본 논문에서는 딥러닝의 이미지 분류 모델에서 악의적 공격자가 생성한 적대적 샘플에 의해 분류 정확도가 낮아짐을 실험적으로 검증하였다. 대표적인 이미지 샘플인 MNIST데이터 셋을 사용하였으며, 텐서플로우와 파이토치라이브러리를 사용하여 만든 오토인코더 분류 모델과 CNN(Convolution neural network)분류 모델에 적대적 샘플을 주입하여 탐지 정확도를 측정한다. 적대적 샘플은 MNIST테스트 데이터 셋을 JSMA(Jacobian-based Saliency Map Attack)방법으로 생성한 방법과 FGSM(Fast Gradient Sign Method)방식으로 변형하여 생성하였으며, 분류 모델에 주입하여 측정하였을 때 최소 21.82%에서 최대 39.08%만큼 탐지 정확도가 낮아짐을 검증하였다.

ABSTRACT

Deep Learning, which is being used in various fields recently, is being threatened with Adversarial Attack. In this paper, we experimentally verify that the classification accuracy is lowered by adversarial samples generated by malicious attackers in image classification models. We used MNIST dataset and measured the detection accuracy by injecting adversarial samples into the Autoencoder classification model and the CNN (Convolution neural network) classification model, which are created using the Tensorflow library and the Pytorch library. Adversarial samples were generated by transforming MNIST test dataset with JSMA(Jacobian-based Saliency Map Attack) and FGSM(Fast Gradient Sign Method). When injected into the classification model, detection accuracy decreased by at least 21.82% up to 39.08%.

**Keywords:** Adversarial attack, MNIST, deep learning, security, autoencoder, convolution neural network

## I. 서 론

오랫동안 연구되어온 인공지능 분야는 최근 인공지능의 난제였던 바둑분야에서 구글이 개발한 '알파고'가 사람을 상대로 이기면서 다시금 활발하게 연구되고 있는 분야이다. 알파고로 인해 유명해진 인공지능의 한 분야인 딥러닝은 여러 분야에서 연구되고 있으며, 보안 분야에서는 스팸메일 탐지, 금융 이상탐지, 악성 코드 탐지 등으로 사용되고 있다. 다른 분야로는 의료분야, 자율주행 자동차 및 인공지능 비

Received(11. 19. 2018), Modified(12. 13. 2018)
Accepted(12. 17. 2018)
* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음(IITP-2018-2016-0-00304)

* 본 연구는 2018년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 이공분야기초연구사업임(NRF-2018R1D1A1B07043349)
† 주저자, qnpfr7179@ajou.ac.kr
‡ 교신저자, tsshon@ajou.ac.kr(Corresponding author)

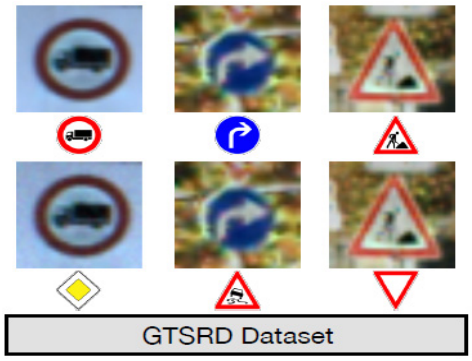


Fig. 1. (top)Original sample (bottom)modified adversarial sample[1]

서 등 범용 적으로 사용되어 생활을 편리하게 해주면서 우리의 삶에 영향력이 커지고 있다.

하지만 지금의 딥러닝 모델은 많은 데이터를 통해 좋은 결과가 나올 때까지 학습을 하여 이루어낸 결과이며, 인간의 뉴런을 흉내 낸 여러 층으로 이루어진 학습 노드 구조로 인해 어떠한 근거로 답을 찾아냈는지는 알기 어렵다. 주어진 문제에 대한 해답을 찾기 위한 과정으로 정답을 찾아낼 만큼의 충분한 데이터를 통해 학습을 한 것이다. 그렇기 때문에 충분한 데이터영역 안에 들어있지 않은 데이터를 입력 값으로 넣었을 경우에는 잘못 판단할 수 있으며 만약 의료분야나 자율주행 자동차와 같은 모델이라면 큰 사고로 이어질 수 있다.

이러한 예는 어렵지 않게 상상할 수 있으며, 만약 자율주행자동차가 Fig.1.과 같이 방향지시 표지판이나 멈춤 표지판을 읽고 판단하려 하였을 때 아래(bottom) 표지판과 같이 악의적 공격자에 의해 변형이 가해진 표지판이었을 경우 사람의 눈에는 정상 표지판처럼 보이지만 딥러닝 모델의 경우 잘 못 판단하여 큰 사고로 이어질 수 있다. 이는 사회 전반의 안전에 대한 위협과 혼란을 야기할 수 있는 것이다. Fig.1.의 아래 그림과 같이 원본 이미지에 변형을 가하여 딥러닝 모델이 잘못 판단하도록 하는 변형된 이미지를 적대적 샘플(adversarial sample)이라고 한다[1].

이처럼 인공지능에 대한 연구와 실생활의 적용이 빠르게 적용되고 확산되는 만큼 딥러닝에 대한 위협 또한 다양해지고 있다. Zhang, Guoming, et al.[2]은 20,000hz 이상의 초음파를 사람이 들을 수 없다는 점을 이용하여 사람의 음성 명령을 사람이 들을 수 없는 초음파 주파수로 변환하여 명령을 실행하도록 하는 돌핀어택(dolphin attack)을 수행하였다. 하지만 국내의 딥러닝 연구는 이러한 적대적 공격에 대한 논의는 충분히 이루어지지 않고 더 나은 학습결과를 위한 연구들이 주를 이루고 있다. 본 논문에서는 이미지 데이터 셋인 MNIST데이터 셋을 JSMA(Jacobian-based Saliency Map Attack)방식과 FGSM(Fast Gradient Sign Method)의 두 가지 방식으로 적대적 샘플을 만들고, 텐서플로우와 파이토치라이브러리를 사용하여 만든 Autoencoder와 CNN(Convolution neural network)분류 모델에 검증 실험을 수행하여 적대적 공격의 위험성을 알아본다.

본 논문의 구성은 다음과 같다. 2장에서는 적대적 샘플의 생성 및 공격에 대한 관련 연구를 소개 하고 3장에서는 적대적 샘플 생성 방법과 딥러닝 분류모델에 대해 서술한다. 4장에서는 사용한 MNIST 데이터 셋에 대한 설명과 적대적 샘플 생성 라이브러리와 학습 모델을 만드는데 사용한 라이브러리에 대하여 설명한다. 5장에서는 생성한 적대적 샘플을 분류모델에 적용하여 적대적 공격의 유효성을 실험하며, 6장에서는 본 논문에 대한 결론과 앞으로 나아가야 할 방향을 제시하고자 한다.

## II. 관련연구

이미지 인식이 최근 중요한 연구 분야로 주목받으면서 딥러닝의 적대적 이미지 공격에 대한 위협은 점차 연구되고 있는 분야이다. 학습모델이 이미지를 판단할 때에는 사진 그 자체를 보는 것이 아니라 픽셀을 이어서 전체의 형태를 파악하여 판단하기 때문에 이어진 선이 어떻게 생겼는가에 따라 판단이 바뀔 수 있다. 만약 픽셀의 위치를 전체적인 그림에 영향이 가지 않게 일부 바꾸어 놓는다면, 딥러닝이 이해하는 전체적인 선의 형태가 달라지면서 잘 못 판단하게 될 확률이 급격하게 높아지는 것이다. Kurakin, Alexey, Ian Goodfellow, and Samy Bengio[3]연구팀은 딥러닝 신경망에 학습을 위해 만들어진 이미지가 아닌 실제 휴대전화의 카메라로 현실의 이미지를 가져온 이미지에 노이즈 필터를 입혀 만든 샘플로 적대적 샘플의 공격을 통해 잘 못 분류될 수 있는지를 보였다. 또한 Samuel G. Finlayson et.al.[4]은 의료 사진에 변형을 가하였을 경우 병을 판단하는 분류 모델에 위협이 될 수 있음을 보이는 연구를 진행하였다. [3]과 [4]처럼 실

제 생활에서 이러한 적대적 샘플에 대한 위험성이나, 적대적 샘플의 생성방법에 대한 여러 선행 연구들이 진행되고 있다.

적대적 샘플을 생성하는 방법으로는 위의 [3], [4]에서 사용한 노이즈를 씌우는 방식으로 Ian Goodfellow, Jonathon Shlens, and Christian Szegedy[5]연구팀에서 제안한 FGSM을 사용하였다. [5]의 FGSM방식은 기존의 이미지를 선형화되도록 하여 모델이 제대로 된 판단을 할 수 없게 하였으며, 원본 이미지와 해당 이미지에서 변화하고자 하는 지점, 변화시킬 크기 변수를 통해 노이즈를 추가하여 생성 시켰다. Papernot et al.[6]에서 연구한 또 다른 적대적 샘플 방법으로는 JSMA방법이 있으며, 입력 값이 출력 값에 미치는 변화를 행렬로 매핑(mapping)하여 분류를 제대로 하지 못하도록 입력 값을 변화시킨다. 이러한 적대적 샘플의 연구는 훈련데이터 셋에 라벨을 올바르게 붙였다고 해서 모델이 해당 분류 작업에 대해 완벽하게 이해를 한다는 것을 의미하지 않음을 보여준다. 적대적 샘플 생성에 대한 FGSM방법과 JSMA방법에 대해 Papernot Nicolas et al.[7]은 텐서플로우에서 동작하는 Cleverhans라이브러리를 제작하였다.

본 논문에서는 [7]에서 제안하는 Cleverhans라이브러리를 사용하여 FGSM방법과 JSMA방법으로 생성한 적대적 샘플과 분류 정확도를 측정하여 딥러닝 모델에 대한 적대적 샘플의 위험성을 실험적으로 알아본다. 국내 딥러닝 연구는 더욱 기존 데이터에 대해 더욱 정확한 결과를 이끌어 내기 위한 연구에 초점이 맞추어져 있으며, 적대적 샘플에 대한 위험성 인식이 부족하다. 공개 딥러닝 라이브러리를 통해 만든 모델을 적대적 샘플에 대한 정확도 감소를 측정하여 적대적 샘플 위험성 인식에 대해 재고한다.

## III. 딥러닝 분류 학습 모델 및 적대적 샘플 생성

### 3.1 딥러닝 분류 학습 모델

이미지를 입력 값으로 사용하여 딥러닝 모델에 학습을 하여 분류를 하기 위한 모델로써, 본 논문에서 사용하는 손 글씨 숫자 이미지를 학습한 모델은 해당 입력 이미지가 어떠한 숫자인지 분류하는 역할을 한다. 본 소절에서는 Autoencoder모델과 이미지 학습 모델인 CNN에 대해 설명한다.

#### 3.1.1 Autoencoder 모델

Fig.2.는 Autoencoder의 기본 형태이며 인코더(encoder)와 디코더(decoder) 부분으로 나누어져 있다. 아래쪽 레이어(layer)에 위치한 $x$는 입력 값을 의미하며 중간 레이어는 히든 레이어(hidden layer)로써 특징(feature)을 추출한다. 입력 값 $x$를 통해 히든 레이어를 거쳐서 출력 값을 재생성 해내는데 재생성한 $x^{'}$는 입력 값 $x$와 유사한 값을 만드는 것을 목표로 한다. Fig.2.에서 아래쪽 레이어의 입력 벡터(vector)인 x는 인코더를 통해 계산되어 히든 레이어의 $y$로 압축되는데 이 식은 (1)과 같다[8]. 이 식에서 $W$는 weight matrix이며, $b$는 bias vector이다. 디코더에서 히든 레이어의 y값은 $x^{'}$으로 다시 매핑 된다. 디코더에서 계산되는 수식 (2)로 디코더를 통해 계산되어 출력 값을 재구성하게 된다.

$$y = s(Wx + b) \tag{1}$$

$$z = s(W^{'}y + b^{'}) \tag{2}$$

위의 두 식에서 $s$는 활성(activation)함수로써 대표적으로는 sigmoid, ReLU(Rectified Linear Unit)등이 있으며, 비용 함수(cost function)의 값이 최소가 되는 지점을 찾기 위해 사용되고 마지막 레이어의 활성함수로 사용되는 함수이다. 히든 레이어에서 최소가 되는 비용함수를 찾아 최적화를 하게 되면 입력 이미지에 대한 특징을 가진다.

분류를 위해서는 학습을 완료 시킨 후 이미지를

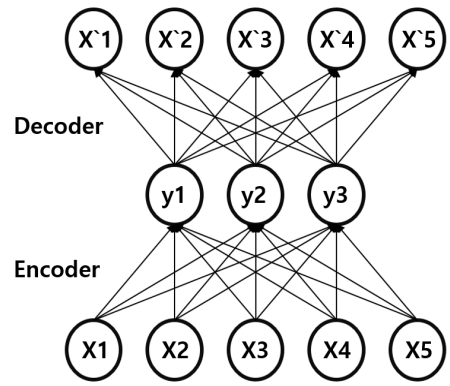


Fig. 2. Basic Autoencoder structure

생성하는 디코더 부분을 제거하고 특징을 학습한 히든 레이어의 $y$값을 가지고 분류를 한다. 분류를 위해서는 여러 번의 학습을 통해 입력 값의 특징을 가지고 있는 히든 레이어를 통해 나온 값에 정답 라벨(label)을 비교하여 분류 함수를 학습한다. 본 논문에서 사용한 분류 함수로는 다항(multinomial)분류인 소프트맥스(softmax) 분류를 사용하여 손글씨 숫자 데이터 셋의 정답 범위인 0~9의 10가지 경우로 분류를 할 수 있다.

### 3.1.2 Convolution Neural Network 모델

CNN모델의 구조는 크게 Convolution 레이어, Pooling 레이어, FC레이어(Fully connected layer)로 나뉠 수 있다. 입력은 이미지이며 구조는 3차원 배열로, 가로(width) x 세로(height) x 색(depth)을 나타낸다. 가로와 세로는 말 그대로 이미지의 크기를 나타내고 세 번째 항은 색이 있는 이미지인지 여부를 나타낸다. 출력은 해당 이미지가 어떤 정답(label)을 가질 확률로 제시되며, FC레이어에서는 이 정보를 이용하여 해당이미지가 결국 어떤 정답을 가질 가능성이 높은지를 나타내주게 된다. 이러한 판단하는 부분을 소프트맥스 분류[9]를 쓰는데, 예측 결과가 여러 개인 경우 사용한다.

Convolution 레이어는 이미지의 특성을 뽑아내는 레이어로서, 입력 이미지에 필터(filter)를 이용하여 이미지의 특성을 뽑아내고 있는데, 여기서 필터는 Fig.3.의 이미지에 있는 빨간색 숫자로 표시된 x 숫자를 의미한다. Fig.3.의 첫 번째 이미지에서 필터는 정확히 1칸씩 움직이며 정보를 추출해내고 있다. 필터를 통해 추출해낸 숫자를 담은 배열을 활성맵(map)이라고 부르고 두 번째 그림인 Convolved Feature이다.

Pooling 레이어는 Sub-Sampling을 하는 것이 목적이며, Convolution을 거친 데이터로부터 한번 더 표본을 뽑는 것을 목표로 한다[10]. Fig.3.의 두 번째 Convolved Feature는 Convolution을 지난 활성 맵이며 여기에서 지역을 나누어 대표 표본을 뽑아내는 것이 Pooling의 목적으로 회색, 노란색, 초록색, 분홍색으로 구역을 나눈다. Fig.3.에서는 Pooling의 방법 중 하나인 Max pooling을 사용했으며, 각 지역에서 가장 큰 값을 뽑아내는 방법이다. Pooling은 다음의 두 가지 효과에 목적이 있다. 첫째, 정말 꼭 필요한 데이터만 뽑아낼 수 있으며, 둘째, 그로 인해 데이터의 양이 작아진다. 이제 마지막으로 결과를 내기 위해 FC레이어에서 이전까지의 정보를 모두 모아 숫자를 예측할 수 있다.

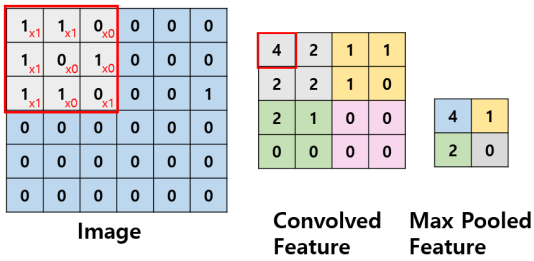


Fig. 3. Convolution process and pooling process

## 3.2 적대적 샘플 생성모델

적대적 샘플은 딥러닝으로 인해 학습된 모델에 공격을 가할 입력 데이터이다. 즉 기존의 정상적으로 잘 분류되는 입력 데이터에서 변형을 일으켜 사람의 눈에는 기존의 데이터와 큰 차이가 없기 때문에 사람은 정확하게 분류할 수 있지만, 딥러닝으로 학습된 모델은 변형으로 인해 잘 못 판단하게 되는 입력 샘플을 생성 혹은 노이즈를 주입해 찾는 것이다. 본 소절에서는 노이즈를 주입하여 변형을 일으킨 샘플을 생성하는 FGSM방법과 JSMA방법에 대해 설명한다.

### 3.2.1 Fast Gradient Sign Method 샘플 생성 방법

FGSM은 Ian Goodfellow et al.[5]에 의해 소개되었다. FGSM의 변형이미지 생성 원리는 수식(3)에서 공격자가 원본이미지 $x$에 딥러닝 모델이 잘 못 판단(분류)하기 원하는 학습지점 $\vec{x}$부근에서, 적대적 공격을 수행할 수 있도록 정의된 모델 $f$를 훈련시키는데 사용된 cost함수 $J$를 선형화하는 것이다. 즉, 제대로 된 학습을 위해서는 입력 값을 넣었을 때 학습 과정에서 비용 함수 $J$를 정답에 가깝도록 출력 값을 변형시켜야 하지만, FGSM방법에선 이를 선형화하도록 학습지점 $\vec{x}$부분을 변형한다. 입력 값 $x$에 대해 적대적 샘플 출력인 $\vec{x}^*$는 다음과 같이 계산된다.

$$\vec{x}^* \leftarrow x + \epsilon \cdot \nabla J(f, \theta, \vec{x}) \tag{3}$$

$\epsilon$은 이미지를 변형시키는 매개변수로써 일종의 노

이즈와 같은 역할을 하며, 그 값이 클수록 모델이 잘못 분류할 가능성이 커진다. 하지만 그만큼 변형이 크게 일어나기 때문에 사람이 더 쉽게 감지할 수 있다.

#### 3.2.2 Jacobian-based Saliency Map Attack 샘플 생성 방법

JSMA(Jacobian-based Sailency Map Attack)생성 방법의 연구는 Papernot et al.[6]에 의해 연구되었다. JSMA방법은 입력 값에 대한 출력 값의 변화를 통해 입력 값의 변화가 정답을 판단하는 출력 값에 미치는 영향을 관찰하여 적대적 샘플을 생성하는 방법이다. 데이터 셋의 입력 값에서 특정 부분을 변형하였을 때 출력의 변동을 매핑(mapping)으로 구성할 수 있는데, 각 히든 레이어의 출력을 다음 히든 레이어의 입력으로 사용하는 것을 매핑하여 최종 출력 값이 정답과 달라지도록 변형시키는 것이다. 수식 (4)는 입력 값 $\vec{x}$의 특징 부분인 $i$와 모델 $f$의 결과 값 부분인 $j$로써 딥러닝 네트워크 함수의 변동을 계산하여 매핑한다.

$$\nabla f(\vec{x}) = \left[ \frac{\partial f_j}{\partial x_i}(\vec{x}) \right]_{i,j} \quad (4)$$

JSMA는 각 특징 부분에 대한 모든 행렬 값을 계산하여 계산의 양이 많을뿐더러, 모든 노드들에 대해 반복적으로 진행하기 때문에 FGSM방법에 비해 계산 비용이 많이 든다.

## IV. 실험 데이터셋 및 라이브러리

본 장에서는 실험에서 사용할 손 글씨 데이터 셋인 MNIST 데이터 셋의 각 파일들이 어떤 파일인지, 이를 어떻게 사용하는지 알아본다. 또한 Autoencoder와 CNN 학습모델 생성에 사용된 텐서플로우와 파이토치 딥러닝 라이브러리가 무엇인지 설명하며, 적대적 샘플 생성 방법인 JSMA방법과 FGSM방법을 사용하기 위한 Cleverhans라이브러리에 대해 알아본다.

### 4.1 MNIST 데이터 셋

MNIST 데이터 셋 필기숫자들이 그레이스케일로 28x28픽셀 이미지로 구성되어져 있으며, 한 자리

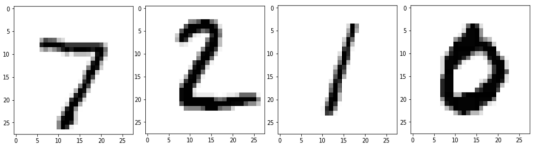

Fig. 4. Part of the MNIST datasets

숫자의 0~9까지의 10가지 종류의 숫자로 이루어져 있다. Fig.4.는 실제 MNIST 데이터 셋의 일부를 이미지화 시킨 것이다. 각각의 그림은 7,2,1,0을 표현하는 이미지 데이터 이다. 파일의 헤더부분인 16 바이트를 읽고 이후 784 (28x28) 바이트가 이미지 파일이며 픽셀 당 1바이트씩 28 x 28 크기로 저장되어 있다. 0에 가까울수록 하얀색에 가까우며 255에 가까울수록 검은색에 가깝다[11].

Table 1.는 MNIST 데이터 셋 파일에 대해 설명한다. 파일 train-images-idx3-ubyte.gz는 ubyte의 파일타입을 gz로 압축시켜놓은 파일로써 여기에는 총 6만개의 이미지 데이터 셋이 들어있다. 5만 5천개의 트레이닝 이미지 데이터와, 5천개의 검증 이미지 데이터가 들어있다. 분류 학습 모델에서는 이 데이터 셋를 사용하여 학습을 할 것이다. 그리고 train-labels-idx1-ubyte.gz는 학습 이미지의 레이블로써 train-images-idx3-ubyte.gz파일에 있는 이미지들의 정답을 가지고 있는 파일이다. t10k-images-idx3-ubyte.gz파일은 테스트 데이터 이미지 셋으로써 1만개의 이미지를 가지고 있으며, 이 테스트 이미지 셋의 정답 레이블은 t10k-labels-idx1-ubyte.gz파일이 가지고 있다. 여기서 테스트 데이터 이미지 셋은 분류 모델의 정확도를 알아보는데 사용하며 본 논문에서는 적대적 샘플과 구분하여 원본(original) 이미지로 지칭한다. 또한 원본 테스트 데이터 이미지 10,000개는 적대적

Table 1. MNIST Dataset Description

| File name | Description |
|---|---|
| train-images-idx3-ubyte.gz | • Training image datasets<br>• 55,000 training images, 5,000 validation images |
| train-labels-idx1-ubyte.gz | • Training image labels |
| t10k-images-idx3-ubyte.gz | • Test image datasets<br>• 10,000 images |
| t10k-labels-idx1-ubyte.gz | • Test image labels |

샘플을 생성하는데 사용하며, 두 가지 방법으로 생성한다. JSMA방법과 FGSM방법을 통해 변형하여 사용한다. 원본 테스트 이미지 셋의 정답 레이블인 t10k-labels-idx1-ubyte.gz파일은 변형한 이미지에서도 같은 정답이므로 적대적 샘플에서도 그대로 사용한다.

### 4.2 딥러닝 학습 라이브러리

딥러닝 학습 모델은 Autoencoder모델과 CNN 모델을 만들었으며 이에 사용된 라이브러리로는 텐서플로우와 파이토치 라이브러리로 구현하였다. 두 가지 라이브러리 모두 같은 파이썬 버전인 3.6버전을 사용하였다.

텐서플로우는 구글에서 만든 라이브러리로서 강화학습(Reinforcement Learning) 및 기타 알고리즘을 지원하는 도구가 있다. 텐서플로우의 목표중 하나는 프로그래머들이 짠 코드를 공유하고 소프트웨어 엔지니어가 딥러닝에 접근하는 방법을 표준화며 누구나 쉽게 어플리케이션을 구현할 수 있는 것을 추구하고 있다[12]. 다른 딥러닝 프레임워크와 마찬가지로 텐서플로우는 C/C++엔진에 파이썬 API로 작성되어 빠른 실행이 가능하다. 하지만 텐서플로우는 실행을 위해서 행렬을 복사해야하며 이는 큰 비용(cost)을 동반하기 때문에 다른 러이브러리 보다 느리다. 본 논문에서는 텐서플로우 버전 1.8.0을 사용하고 있다.

파이토치는 다중 패러다임 스크립팅 언어인 Lua로 작성된 API로 만들어진 라이브러리로써 다양한 버전이 존재한다. 본 논문에서는 파이토치를 정식으로 지원하는 가장 최신버전인 0.3.1.post2를 사용하였다. 파이토치는 페이스북에서 오픈 소스화 된 딥러닝 라이브러리로써 파이썬 API로 작성되어 있다. 이는 가변길이 입력 및 출력을 처리할 수 있는 동적 계산(dynamic computation)그래프를 제공하며, 특히 시변적 특징을 가지는 데이터를 처리하는 RNN(recurrent neural network)모델을 사용할 때 유용하다. 많은 모듈로 구현되어 있으며, GPU로 실행하기 쉽다.

### 4.3 적대적 샘플 생성 라이브러리

본 논문에서는 텐서플로우로 구현되어 제공하는 Cleverhans라이브러리[7]을 사용하여 적대적 샘플을 생성하였으며, FGSM방법과 JSMA방법을 사용하였다. 이 라이브러리는 구글 딥마인드 팀과 Pennsylvania 대학이 협력하여 텐서플로우를 사용하여 구현하였기 때문에 텐서플로우 환경에서 사용할 수 있다. 3절에서 기술한 JSMA방법과 FGSM방법으로 원본 이미지에 변형을 가하는 방법을 제공하고 있다. 또한 GPU사용을 지원하기 때문에 GPU버전의 환경을 구축하는 것이 성능 향상에 좋으며, 라이브러리 개발은 Ubuntu환경에서 진행하지만 윈도우 버전과 OSX에서도 동작가능 하다. 본 논문에서는 텐서플로우를 사용한 학습모델에서 FGSM로 변형한 적대적 샘플의 이미지를 생성한 후, 텐서플로우와 파이토치 라이브러리로 제작한 학습모델에 대한 오학습을 실험한다. 2018년 5월 초에 v2.1.0 버전이 업데이트 되었으며, 실험을 진행 중일 때 최신버전은 v2.0.0으로 해당 버전으로 실험을 진행하였다.

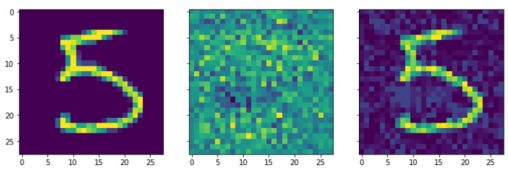

Fig. 5. Adversarial sample using FGSM

## V. 실험 방법 및 결과

### 5.1 실험환경 및 방법

실험환경으로는 Table 2.와 같다. 인텔 CPU i7-8700 3.19GHz와 16G램 그리고 GPU GTX 1060 6G를 사용하였으며, 운영체제 Window 10 Pro 64bit에서 아나콘다 가상환경에서 파이썬 3.6 버전으로 구성하였다.

모델 생성에는 텐서플로우와 파이토치 두 가지 라이브러리로 Autoencoder와 CNN 학습모델을 만들어 MNIST데이터 셋의 손글씨 분류를 진행하였다. 우선 구글에서 제공하는 MNIST 데이터 셋으로 학습하여 분류 정확도를 측정한 후, JSMA방법과 FGSM방법으로 만든 적대적 샘플을 적용하여 CNN학습모델과 Autoencoder학습모델의 분류 정확도가 얼마나 감소하는지 측정하였다. Fig.5.는 FGSM으로 만들어진 적대적 샘플의 한 예로써, 좌측의 원본 그림에 중간의 노이즈를 주입하여 오른쪽의 적대적 샘플을 만들어낸다.

Table 2. Experiment Environment

| Categorization | Specification |
|---|---|
| CPU | i7-8700 3.19GHz |
| RAM | 16G |
| GPU | GTX 1060 6G |
| OS | Window 10 Pro 64bit |
| Python | 3.6 version |
| Tensorflow | 1.8.0 version |
| Pytorch | 0.3.1.post2 version |

본 실험에서는 분류 정확도(Accuracy)를 통해 모델의 분류 정확도를 측정할 것이며, 분류 정확도는 추론된 결과가 참이며 실제 결과도 참인 TP(True Positive), 추론된 결과는 참이나 실제 결과는 거짓인 FP(False Positive), 추론된 결과는 거짓이나 실제 결과는 참인 FN(False Negative) 그리고 추론된 결과가 거짓이며 실제 결과도 거짓인 TN(True Negative)으로 수식 (4)로 계산을 하며 단위는 %이다.

$$\text{정확도}(Accuracy) = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

### 5.2 실험 및 결과

텐서플로우 라이브러리로 구현된 Autoencoder 모델은 2개의 히든 레이어로 구성되어 있으며, 각 히든 레이어 노드의 개수는 200개이다. 학습 epoch는 200, 즉 학습 반복 횟수는 200번으로 하였으며 활성 함수는 sigmoid함수를 사용하였다. 레이어와 노드의 개수, epoch와 반복 횟수는 기본 데이터 셋에 대해 90%이상의 분류 정확도를 보이며 간단하게 구성할 수 있도록 반복적 실험을 통해 설정한 값이다. 분류는 소프트맥스 분류 방법으로 하였으며, 기본 소프트맥스인 tf.nn.softmax 로 사용하였다. Learning rate와 Gradient Descent rate는 일반적으로 많이 사용하는 값 중 하나인 0.01와 0.5로 설정하였다. 이 두 값은 특별한 수학적 의미가 있는 것은 아닌 많은 학습결과를 통해 경험적으로 최적화된 값을 사용하였다. 이러한 변수들의 값 설정은 파이토치에서도 동일하게 설정하였으며, 각 라이브러리의 동일한 환경의 분류 측정을 위해 최대한 커스텀하지 않고 라이브러리에서 제공하는 기본적인 함수를 사용할 것이다. CNN 모델의 경우에도 2개 계층의 convolution을 적용하였으며, Autoencoder와 마찬가지로 소프트맥스 함수를 사용하였다. Convolution 필터의 이동정도를 의미하는 strides는 각각의 픽셀을 하나의 입력 값으로 사용하기 위해 (1,1,1,1)로 하였다.

텐서플로우 라이브러리에서 Autoencoder모델의 분류 정확도는 구글에서 제공하는 기본 MNIST 원본(original)테스트 데이터 셋의 경우 Table 3.과 같이 96.35%가 나왔으며, 생성한 적대적 샘플의 경우 FGSM샘플은 54.27% 그리고 JSMA샘플은 66.13%로 분류되었다. CNN모델의 경우 Original 데이터 셋은 99.17%, FGSM샘플은 58.13% 그리고 JSMA샘플은 71.21%로 측정되어 Autoencoder모델보다는 높은 수치의 분류 정확도를 보여주었다.

파이토치 라이브러리의 모델들도 텐서플로우와 마찬가지로 아나콘다 가상환경을 사용 하였으며, python=3.6으로 작성하였다. 분류 정확도의 결과에 일관성을 가지기 위해 epoch를 텐서플로우 라이브러리를 사용한 딥러닝 모델들과 마찬가지로 200으로 하였고 learning rate 또한 0.01로 할당하였다. 실험 결과는 Table 4.과 같이 Autoencoder모델에서 원본 테스트 데이터 셋의 경우 92.72%의 분류 정확도로 측정되었으며 FGSM샘플은 62.23% 그리고 JSMA샘플은 70.76%의 분류 정확도를 보였다. 하지만 CNN모델의 경우 원본 테스트 데이터 셋의 분류 정확도는 99.29%로 높은 분류 정확도를 보여주었으며 FGSM샘플은 73.23% 그리고 JSMA샘

Table 3. Tensorflow Library Accuracy(%)

| Model / Data type | Autoencoder | CNN |
|---|---|---|
| Original | 96.35 | 99.17 |
| FGSM | 57.27 | 65.13 |
| JSMA | 66.13 | 71.21 |

Table 4. Pytorch Library Accuracy(%)

| Model / Data type | Autoencoder | CNN |
|---|---|---|
| Original | 92.72 | 99.29 |
| FGSM | 62.23 | 71.23 |
| JSMA | 70.76 | 77.47 |

플은 77.47%의 분류 정확도를 보였다.

## VI. 결론 및 향후연구

딥러닝 모델에 대한 다양한 취약성이 있지만, 본 논문에서는 알려진 샘플에 대해 변형을 가한 데이터로 분류 정확도를 측정하여 적대적 샘플을 통한 오학습 및 탐지 정확도의 감소를 실험적으로 검증하였다. 이미지 데이터 셋에 제한 받지 않는 Autoencoder분류모델과 이미지에 특화된 CNN분류모델에, 기존 MNIST 데이터로 FGSM방법과 JSMA방법으로 생성한 적대적 샘플을 분류 모델에 테스트 셋으로 사용하여 분류 정확도를 측정하였다.

FGSM방법으로 생성한 샘플은 텐서플로우로 작성한 Autoencoder분류모델에서 42.08% 감소하였고 JSMA샘플은 30.22% 감소하였다. CNN분류모델에서는 FGSM샘플 또한 34.04% 감소하였으며, JSMA샘플은 27.96% 감소하였다. 이는 이미지 데이터 셋인 MNIST가 이미지 기반의 CNN분류 모델이 범용적으로 사용될 수 있거나 혹은 특징 추출에 주로 사용되는 모델인 Autoencoder분류 모델에 비해 분류 정확도가 높음을 보여주지만, 두 분류 모델 모두 적대적 샘플에 의해 분류 정확도가 감소하는 것을 알 수 있었다. 파이토치로 작성한 분류 모델도 Autoencoder분류 모델의 경우 FGSM샘플은 92.72%에서 62.23%로 감소하였고, JSMA샘플은 70.76%로 감소하였다. 또한 CNN분류 모델은 FGSM샘플의 경우 99.29%에서 73.23%로 감소하였으며, JSMA샘플은 77.47%로 감소하였다. 텐서플로우 라이브러리로 작성한 분류모델과 비교하였을 때, 전체적인 분류 정확도에서 파이토치 라이브러리로 작성한 분류 모델이 더 높은 분류 정확도를 보여준다. 이러한 결과는 같은 기능을 하는 함수도 라이브러리별 구현하는 방식에 따라 정확도 및 성능에 영향을 미치는 것을 알 수 있다.

위의 실험과 같이 적대적 샘플에 의해 쉽게 오 학습 및 탐지 정확도가 낮아지는 것을 알 수 있으며, 딥러닝을 활용하여 병의 진단, 네트워크 침입 탐지 그리고 자율주행 자동차등 실생활에 밀접한 곳에 사용되고 있는데 적대적 샘플이 악용될 경우 큰 피해를 입을 수 있다. 이러한 적대적 공격을 방지하기 위해서는 원본 이미지를 변형시킨 적대적 샘플들을 재학습 시켜 오탐률을 줄일 수 있다. 근본적인 방법으로는 워터마크 혹은 신뢰성 있는 인증된 데이터 셋만을 분류 모델의 입력 값으로 사용하는 방법이 있으나, 모든 데이터 셋을 검증하기란 쉽지 않다.

향후 연구에서는 기존 실험에 DeepFool방법의 적대적 샘플 생성과 RNN분류 모델등의 학습방법 및 라이브러리를 추가적으로 확장하여 다양한 환경에 범용적인 적대적 샘플 위험성을 연구할 것이며, 적대적 샘플을 재학습하여 강건한 모델을 구축하는 연구를 진행할 것이다.

## References


[1] Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, pp. 506-519, Apr. 2017.

[2] Zhang, Guoming, et al. "Dolphin Attack: Inaudible voice commands." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp.103-117, Oct. 2017.

[3] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." arXiv preprint arXiv:1607.02533. 2016.

[4] Finlayson, Samuel G., Isaac S. Kohane, and Andrew L. Beam. "Adversarial Attacks Against Medical Deep Learning Systems." arXiv preprint arXiv:1804.05296 .2018.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[6] Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. IEEE, pp.372-387, Mar. 2016.

[7] Papernot, Nicolas, et al. "cleverhans v2. 0.0: an adversarial machine

learning library." arXiv preprint arXiv:1610.00768 . 2016..
〔8〕 Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of machine learning research 11. pp.3371-3408, Dec. 2010.
〔9〕 Heckerman, David, and Christopher Meek. "Models and selection criteria for regression and classification." Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc, pp.223-228, Aug. 1997.
〔10〕 Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems, pp.1097-1105. 2012.
〔11〕 Bottou, Léon, et al. "Comparison of classifier methods: a case study in handwritten digit recognition." Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on. Vol. 2. IEEE, pp.77-82, Oct. 1994.
〔12〕 Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. pp.265-283, Nov. 2016.

## 〈저자소개〉

정 재 한(JaeHan Jeong) 학생회원
2017년: 아주대학교 정보컴퓨터공학과 졸업(학사)
2017년~현재: 아주대학교 컴퓨터공학과 재학(석사)
〈관심분야〉 침입탐지시스템, 딥러닝, IoT 보안

손 태 식 (Taeshik Shon) 종신회원
2000년: 아주대학교 정보및컴퓨터공학부 졸업(학사)
2002년: 아주대학교 정보통신전문대학원 졸업(석사)
2005년: 고려대학교 정보보호대학원 졸업(박사)
2004년~2005년: University of Minnesota 방문연구원
2005년~2011년: 삼성전자 통신· DMC 연구소 책임연구원
2017년~2018년: Illinois Institute of Technology 방문교수
2011년~현재: 아주대학교 정보통신대학 사이버보안학과 교수
〈관심분야〉 ICS/SCADA, DFIR, Anomaly Detection