# COSE474-2023F: Final Project
# PGD: Advanced Adversarial Attack of FGSM

사이버국방학과 2020330003 최창호

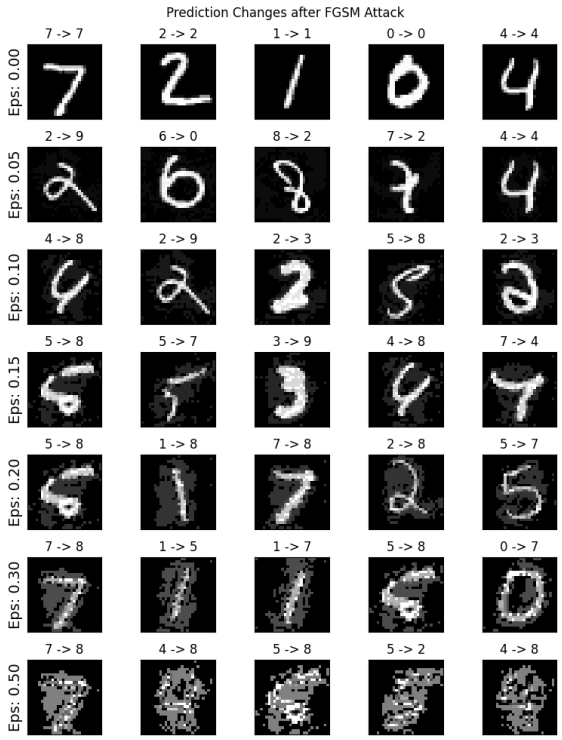


*Figure 1.* Model Predictions after FGSM Attack

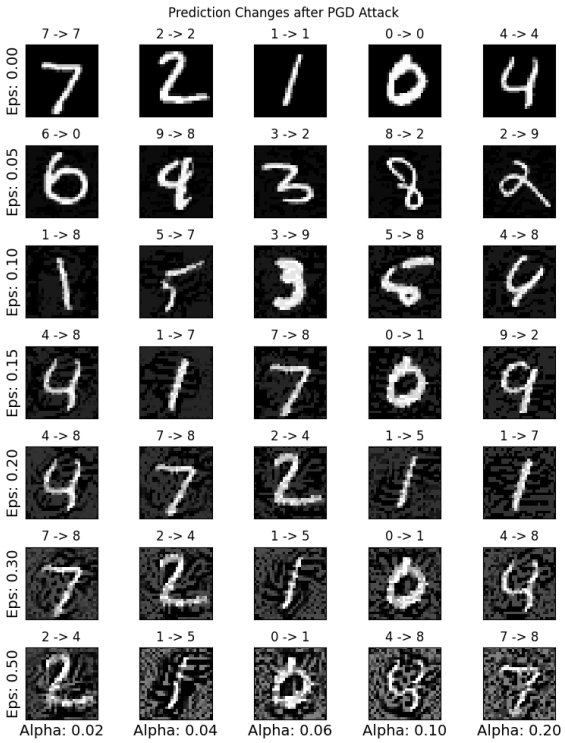


*Figure 2.* Model Predictions after PGD Attack

## 1. Introduction

최근 AI 활용과 관련하여 폭발적으로 늘어난 관심에 비해 AI Security 및 Attack에 관한 주목도는 비교적 적다. 그러나 AI의 특성상 사용자가 AI model이 어떠한 근거를 가지고 결과가 도출되었는지를 알기 어렵기 때문에 공격을 당해도 공격을 당한 것을 확인하기가 어렵다. AI Adversarial Attack의 시작이라고 할 수 있는 Goodfellow의 Fast Gradient Sign Method(FGSM)(Goodfellow et al., 2015)에서는 Adversarial Attack의 가능성을 제시하였고, 이후 FGSM을 개선한 Projected Gradient Descent(PGD) Attack(Madry et al., 2019)이 제시되었다. 이에 본 프로젝트에서는 PyTorch에서 제공하는 FGSM의 Skeleton Code를 활용하여 PGD를 구현해보고 성능과 결과를 분석하며, 최종적으로 PGD의 효과를 제시함으로써 AI Security에 대한 관심을 높이고자 한다.

## 2. Motivation & Problem definition

본 프로젝트에서는 두 가지 관점에서 연구를 진행하였다. 첫번째는 PGD의 기본이 되는 FGSM의 수학적 증명이고, 두 번째는 실제로 PGD의 Targeted Attack을 구현하여 FGSM과의 비교를 통해 어느 정도 성능 향상을 이루었는지 확인하는 것이다. 또한 PGD의 주요 Hyperparameter인 Alpha와 Epsilon의 변화에 따른 성능 비교도 진행한 후 공격 성능의 직관적인 파악을 위하여 공격 이후의 이미지를 시각적으로 출력하여 결과를 확인하였다.

## 3. Background

FGSM과 PGD Attack은 Input Data에 적절한 Perturbation(변조)를 줌으로써 인간이 보기에는 큰 변화가 없어 정상적인 데이터로 보이지만 Model은 원래의 결과와 다른 결과를 출력하도록 하는 공격 기법이다. 이때

공격은 Model이 기존의 Label과 다른 결과만 나타내기만 하면 공격 성공으로 보는 Untargeted Attack과 Model이 공격자가 원하는 Target으로 판단하도록 하는 Targeted Attack으로 구분된다. 본 프로젝트에서는 Targeted Attack을 기준으로, FGSM과 PGD의 수학적, 이론적 배경을 살펴보고, 실제 공격 결과를 분석하였다.

FGSM에서 Targeted Attack의 경우 데이터 x에 가해진 perturbation을 r, 이후의 결과를 x'=x+r라고 하였을 때, 공격이 성공하기 위한 이상적인 r*과 이때의 x'은 다음과 같은 식을 만족한다.

$$x^{'} = x - \epsilon * sign(\nabla_x l(x,t))$$

이때 t는 x에 대해 Attacker가 원하는 결과 target t, l은 x와 target t사이의 loss function이고고, sign 함수는 다음과 같다.

$$sign(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & if\ x = 0 \\ -1 & if\ x < 0 \end{cases}$$

## 4. Method

FGSM은 수학적으로 도출된 결과이지만, 수학적으로 증명하는 과정에서 perturbation r을 구할 때 테일러 급수를 사용하였다. 따라서 r* = x+r에서 r은 큰 값을 가지면 안되며 그렇기 때문에 큰 perturbation을 줄 수 없다. 이를 해결하고자 PGD가 제시되었다.

PGD는 FGSM을 여러 iteration 수행하되 각 iteration의 결과가 처음 이미지 x와의 차이가 $\epsilon$ 를 넘지 않도록 하며 진행하는 공격 방법이다. $\epsilon$ 으로 제한을 두는 이유는 iteration이 반복되면서 너무 많은 perturbation이 가해지면 데이터가 변조되었다는 것이 육안으로 확인되어 공격이 발각되기 때문이다. 결과적으로 PGD는 다음과 같은 식을 통해 해를 구할 수 있다.

$$x^{i+1} = Clip_{x,\epsilon}\{x^i - \alpha * sign(\nabla_x l(x,t))\}\, for\, i = 0 \dots p$$

이때 Clip 함수는 다음과 같다.

$$Clip_{X,\epsilon}(x) = \min\{X + \epsilon, \max\{x, X - \epsilon\}\}$$

Clip 함수를 통해 Perturbed의 정도를 제한할 수 있다.

## 5. Mathematical Proof of FGSM's Optimal Solution

다음과 같은 식을 만족하는 최적해 r*을 찾자.

$$r^* \in \arg\min_{r \in \mathbb{R}^p} L(x + r^*, t)$$

이때 테일러 급수를 사용하면 r*은 다음과 같이 근사할 수 있다.

$$r^* \in \arg\min_{r \in \mathbb{R}^p} L(x,t) + r^T \nabla_x L(x,t)$$

$$\text{subject to: } \|r\|_\infty \le \epsilon, \quad i = 1, \dots, p$$

$\mathbf{L}(x,t)$ 부분은 $r$과 관련 없는 부분이므로 최적해를 구하는 과정에서는 무시할 수 있다. 따라서 우리가 원하는 r*은 다음 조건을 만족하는 경우이다.

$$\min_{r \in \mathbb{R}^p} r^T \nabla_x L(x,t)$$

$$\text{subject to: } \begin{cases} \epsilon - r_i \ge 0, & i = 1, \dots, p \\ \epsilon + r_i \ge 0, & i = 1, \dots, p \end{cases}$$

Lagrangian function과 Loss function을 구분하기 위하여 각각 $L$과 $l$로 표현하였다. The Lagrangian function is:

$$L(r, \alpha, \beta) = r^T \nabla_x l(x,t) - \sum_{i=1}^{p} \alpha_i(\epsilon - r_i) - \sum_{i=1}^{p} \beta_i(\epsilon + r_i)$$

KKT conditions:

$$\begin{cases} Stationary: \ [\nabla L(r,\alpha,\beta)]_i = [\nabla_x l(x,t)]_i + \alpha_i - \beta_i = 0 \\ Primal\, Feasibility: \ -\epsilon \le r_i \le \epsilon \\ Dual\, Feasibility: \ \alpha_i \ge 0, \beta_i \ge 0, \quad i = 1, \dots, p \\ Complementarity: \ \alpha_i(\epsilon - r_i) = 0, \ \beta_i(\epsilon + r_i) = 0 \end{cases}$$

여기서 r과 $\alpha$, $\beta$ 사이의 다음과 같은 관계식을 구할 수 있다.

$$\begin{cases} if\ |r_i| < \epsilon, then\ \alpha_i = 0, \beta_i = 0 : [\nabla_x l(x,t)]_i = 0 \\ if\ r_i = \epsilon, \ then\ \beta_i = 0 : [\nabla_x l(x,t)]_i = -\alpha_i \le 0 \\ if\ r_i = -\epsilon, \ then\ \alpha_i = 0 : [\nabla_x l(x,t)]_i = \beta_i \ge 0 \end{cases}$$

따라서 $r_i$는 loss function의 gradient만으로 구할 수 있다.

$$r_i \quad = \quad \begin{cases} +\epsilon & if\ [\nabla_x l(x,t)]_i < 0 \\ -\epsilon & if\ [\nabla_x l(x,t)]_i > 0 \\ 0 & o.w. \end{cases}$$

정리하면 FGSM Targeted Attack은 다음과 같은 최적해를 갖는다.

$$x^{'} = x - \epsilon * sign(\nabla_x l(x,t))$$

## 6. Baseline & Datasets

PGD가 FGSM을 발전시킨 모델인만큼 FGSM을 Baseline으로 잡아 성능 비교를 진행하였다. Dataset으로는 MNIST Dataset을 사용하였는데, MNIST Dataset은 size가 크지 않아 적은 resource로도 돌릴 수 있으며 결과를 직관적으로 확인할 수 있기 때문에 데이터셋으로 결정하였다.

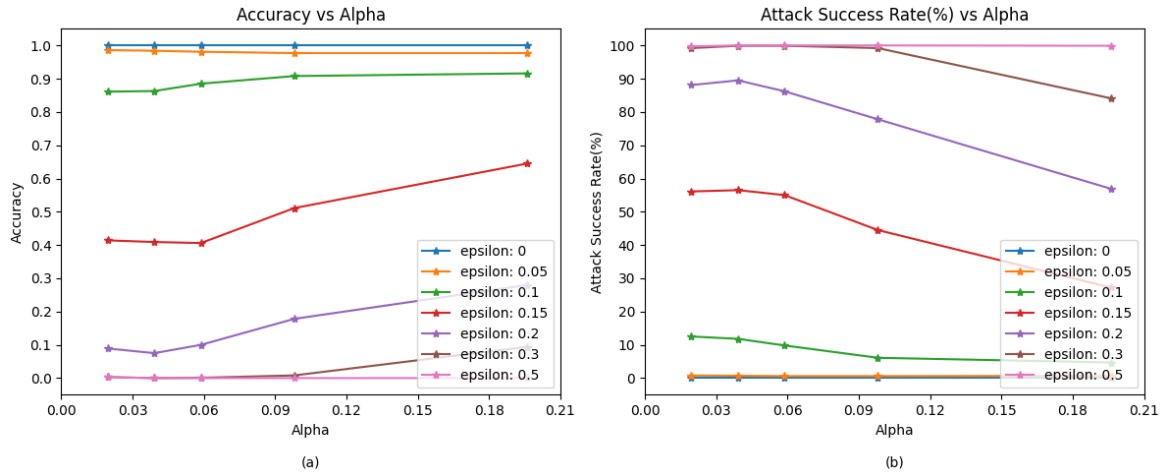


*Figure 3.* Effect of Alpha in PGD Attack

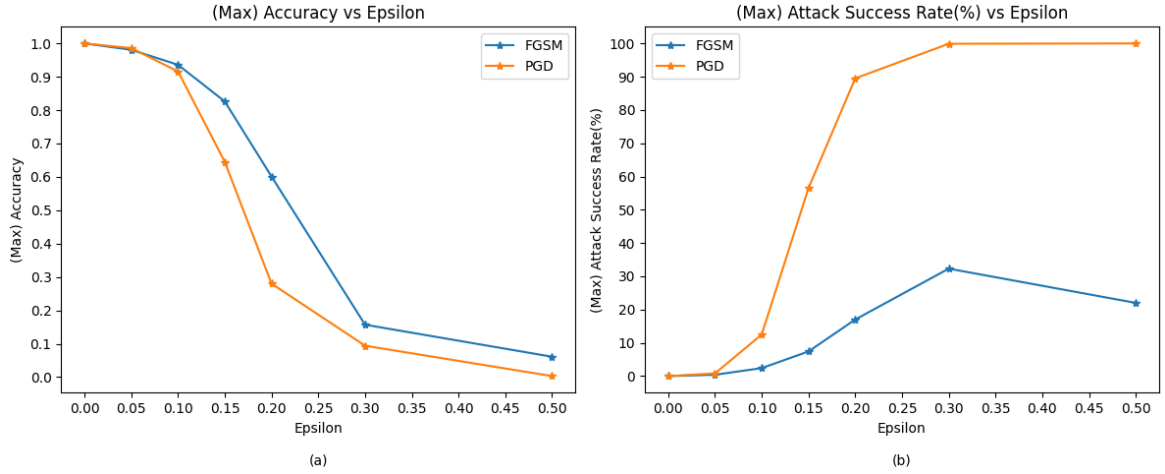


*Figure 4.* Comparison between FGSM and PGD Attacks

## 7. Experiments

실험은 Google Colab T4 GPU를 사용하여 진행하였다. FGSM과 PGD의 Victim에 해당하는 기본 AI Model은 MNIST Dataset으로 Pretrained된 lenet을 사용하였다. PGD의 iteration은 25로 고정하였고, epsilon과 alpha 값에 변화를 주며 결과값을 비교하였다.

성능은 Accuracy와 Attack Success Rate 두 가지로 측정하였다. Accuracy는 Perturbation을 준 Data들에 대해 기존 model이 어느 정도의 Accuracy를 보이는지, Attack Success Rate는 해당 Data들에 대해 model이 Attacker의 Target으로 판단한 수의 비율을 의미한다. 같은 Attack Success Rate라면 Accuracy가 높은 것이, 같은 Accuracy라면 Attack Success Rate 높은 것이 공격이 효과적으로 진행된 것으로 해석할 수 있는데, Accuracy가 떨어지면 사용자가 이상을 발견할 확률이 높아지기 때문이다. Figure 3, 4를 살펴보면 Accuracy와 Attack Success Rate는 반비례 관계인 것을 확인할 수 있다. Perturbation을 많이 줄수록 Attack이 성공할 확률이 높아지지만 그만큼 이미지에 변조가 많이 되기 때문에 Model의 Accuracy가 떨어지는 것이다. 또한 $\epsilon$이 증가하면 Accuracy는 떨어지고 Attack Success Rate는 증가하는데, 이것은 $\epsilon$이 Perturbation되는 정도와 직결되는 Hyperparameter이기 때문이다.

### 7.1. Effect of Alpha in PGD Attack

Figure 3를 살펴보면 PGD의 $\alpha$의 값에 따라 성능이 바뀌는 것을 확인할 수 있다. Monotonic Function은 아니지만 비슷한 경향성을 보인다. $\alpha$가 너무 커졌을 때 오히려 성능이 감소하는 이유는 PGD의 수식에서 $\alpha$는 FGSM에서의 $\epsilon$과 같은 역할이며, 즉 큰 값을 가지면 안되기 때문이다. $\epsilon$이 충분히 커지면 Attack Success Rate가 100%에 가깝게 수렴하는 것도 볼 수 있다.

### 7.2. Comparison of FGSM and PGD Attacks

Figure 4는 같은 $\epsilon$ 값을 사용하였을 때 FGSM과 PGD의 성능을 비교한 것이다. PGD의 성능은 $\alpha$의 값에 영향을 받기 때문에 여러 $\alpha$의 결과 중 Maximum값으로 비교하였다. (a)를 보면 Accuracy는 PGD가 FGSM에 비해 떨어진다.

반면 (b)를 살펴보면 Attack Success Rate에서는 크게 앞서는데, 특히 $\epsilon$ 커지면서 성공 비율이 큰 폭으로 차이가 나는 것을 볼 수 있다. 이는 앞서 언급했던 FGSM이 갖고 있는 테일러 급수를 사용했다는 근본적인 문제 때문이다. 테일러 급수는 작은 perturbation에서만 적용할 수 있기 때문에 $\epsilon$ 커짐으로써 큰 perturbation을 사용하게 되면 Optimal Solution 자체가 아니게 되기 때문이다. 심지어 FGSM에서는 $\epsilon$이 일정 크기 이상 커졌을 때 오히려 Attack Success Rate가 떨어지는 것을 볼 수 있었다. 이것이 PGD의 효과를 가장 정확히 보여준다.

### 7.3. Qualitative Results

Figure 1과 2는 각각 FGSM과 PGD Attack 이후의 Pertubed Data와 그때의 Model output을 나타낸 그림이다. $\epsilon$이 0일 때는 Perturbation이 가해지지 않은 상태이므로 원본 그대로이고 Model도 정확하게 예측한다. $\epsilon$이 커질수록 이미지에 가해지는 Perturbation이 커지는 것을 시각적으로 확인할 수 있다. 또한 $\epsilon$이 작은 값들의 경우 Perturbation이 가해진 것이 육안으로는 확인되지 않지만 Model은 Attacker의 의도대로 Target으로 판단한다. 나아가 Figure 4를 보면 $\epsilon$이 커졌을 때 Accuracy는 극단적으로 작아지는데, 실제로 Figure 1과 2를 통해 $\epsilon$이 커졌을 때 Data가 육안으로도 파악이 불가능할 정도로 perturbed되는 것을 확인할 수 있다.

## 8. Further Discussion

연구를 진행하며 몇 가지 Discussion Points를 발견할 수 있었다.

### 8.1. Consider to non-target classes

첫 번째 Discussion Point는 non-target class들에 대한 것들도 Loss에 포함시키는 것이다. 기존의 PGD는 target 간의 loss만 계산하고 label space 내의 다른 non-target class들에 대한 것은 고려하지 않았기 때문이다. non-target class들까지 loss에 포함시킨다면 같은 Attack Success Rate일 때 Accuracy가 훨씬 높아질 것으로 기대된다.

### 8.2. Attack Scenarios

두 번째 Point는 그래서 이 공격을 어떻게 활용할 것인지이다. PGD Targeted Attack이 시사하는 바는 Model이 Attacker가 원하는 Target으로 판단하도록 유도할 수 있다는 것이다. 이것을 악용한다면 여러 가지 경우에 사용할 수 있는데 에를 들어 여권 사진이나 주민번호증처럼 개인 사진을 보고 신원을 파악할 때에 사진에 적절한 Perturbation을 줌으로써 다른 사람으로 인식하게 하거나 자동차 번호판을 다른 번호로 보이게 하는 등이 있을 수 있다. 물론 이 공격은 Model을 알아야 가능하다는 한계점이 존재한다.

## 9. Conclusion

본 프로젝트에서는 FGSM에서 발전한 형태인 PGD를 직접 구현해보고 그 성능을 확인해보았다. PGD를 사용함으로써 Accuracy를 어느정도 보장하는 선에서 Data를 Attacker가 원하는 방향으로 Perturbed 시킬 수 있었고 결과를 시각적으로 분석하였다. Adversarial Attack의 기본에 해당하는 기법이지만 직접 구현해보고 분석하면서 Adversarial Attack의 효과를 제대로 확인할 수 있었다고 생각한다. AI의 중요성이 갈수록 커지고 있는만큼 Adversarial Attack을 비롯한 AI Security에 보다 많은 관심이 생겼으면 좋겠다.

## References

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks, 2019.

## A. Links

Github Code: https://github.com/hoho4702/DeepLearning-Final-Project

Skeleton Code: https://pytorch.org/tutorials/beginner/fgsm_tutorial.html

Dataset: https://github.com/pytorch/examples/tree/main/mnist