Deep Learning

2025 Summer

Yangjun Ahn
yangjunahn@sungshin.ac.kr
https://sites.google.com/sungshin.ac.kr/mhail

GAN

- Adversarial Example Generation
- DCGAN Tutorial

Introduction – Adversarial Example Generation

- ML 모델의 보안
  - 개발 방향은 주로 빠르고, 정확하고, 효율적으로
  - 모델을 속이려는 적에 대한 보안과 견고함은?
- 위협 모델
  - 이미지에 작은 변화(perturbation)를 주어 성능을 떨어뜨리는 방법
  - 정보에 따라:
    - 화이트박스
    - 블랙박스

위협 모델

- 화이트박스: 공격자가 모델의 입력, 출력, 가중치 등 모든 정보를 알고 있는 경우를 가정
- 블랙박스: 공격자가 입력, 출력만 알고 있는 경우를 가정
- 예시: Fast Gradient Sign Method (FGSM)
  - https://arxiv.org/abs/1412.6572
  - 화이트박스 공격
  - 오분류(분류 결과를 틀리게 하고, 어떻게 분류되는지는 관심 없음) 목표

Fast Gradient Sign Method (FGSM)

- 화이트박스 공격이므로 구조와 가중치를 알고 있다고 가정
- 방법
  - 입력 데이터에서 계산한 손실 변화도를 사용, 역전파 변화도를 최대가 되게 조정하여 손실을 최대가 되도록 만드는 방법

https://arxiv.org/abs/1412.6572

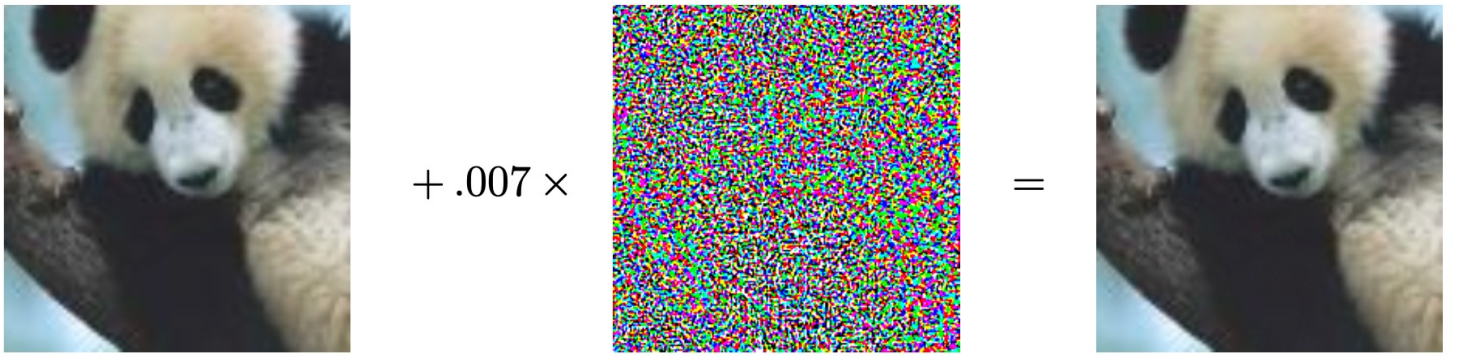
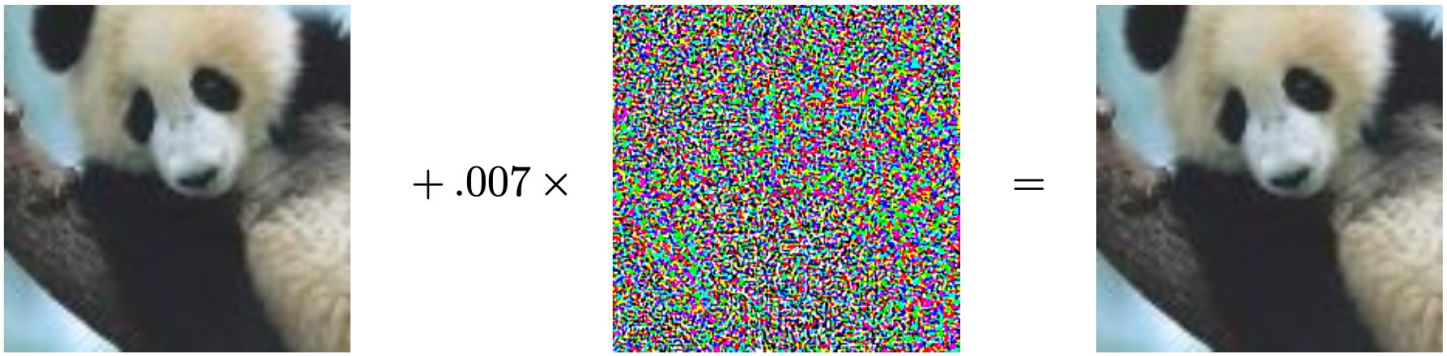


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet’s classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet’s conversion to real numbers.

Fast Gradient Sign Method (FGSM)



$\boldsymbol{x}$ $\quad$ $\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ $\quad$ $\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

$$\boldsymbol{\eta} = \epsilon \, \text{sign} \, \nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y)$$

- $\boldsymbol{x}$ the input to the model, y the targets associated with , $J(\boldsymbol{\theta}, \boldsymbol{x}, y)$ the cost used to train the neurla network.
- {perturbed image} = {image} + epsilon × sign {gradient of the loss w.r.t the input image}

FGSM Hand Practice

- 클래스: 개(0), 고양이(1), 말(2)
- 입력: $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \mathbf{2.0} \\ \mathbf{1.0} \end{bmatrix}$, 정답: $y = [1.0]$ (고양이)
- 학습 완료 모델: $\mathbf{w} = \begin{bmatrix} 1.0 & -1.0 \\ 0.5 & 1.0 \\ -1.0 & 0.0 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 0.0 \\ 0.5 \\ -0.5 \end{bmatrix}$
- 순전파(feedforward):

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} = \begin{bmatrix} 1.0 \\ 2.5 \\ -2.5 \end{bmatrix}$$

$$\mathbf{softmax}(\mathbf{z}) \cong \begin{bmatrix} 0.080 \\ 0.918 \\ 0.002 \end{bmatrix}$$

FGSM Hand Practice

$$\boldsymbol{\eta} = \epsilon \operatorname{sign} \nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y)$$

- 손실 기울기 계산

$$\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y) = \mathbf{W}^T(\hat{y} - y) = \begin{bmatrix} 1.0 & 0.5 & -1.0 \\ -1.0 & 1.0 & 0.0 \end{bmatrix} \begin{bmatrix} 0.080 - 0.0 \\ 0.918 - 1.0 \\ 0.002 - 0.0 \end{bmatrix} = \begin{bmatrix} 0.037 \\ -0.162 \end{bmatrix}$$

- FGSM Perturbation with $\epsilon = 0.7$
  - Perturbed image:

$$\mathbf{x} + \boldsymbol{\eta} = \mathbf{x} + \epsilon \operatorname{sign} \nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y) = \begin{bmatrix} 2.0 \\ 1.0 \end{bmatrix} + 0.7 \times \begin{bmatrix} 1.0 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 2.7 \\ 0.3 \end{bmatrix}$$

FGSM Hand Practice

- 공격한 이미지를 이용한 순전파(feedforward):

$$\mathbf{z} = \mathbf{W} \cdot (\mathbf{x} + \boldsymbol{\eta}) + \mathbf{b} = \begin{bmatrix} 1.0 & -1.0 \\ 0.5 & 1.0 \\ -1.0 & 0.0 \end{bmatrix} \cdot \begin{bmatrix} 2.7 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.0 \\ 0.5 \\ -0.5 \end{bmatrix}$$

$$= \begin{bmatrix} 2.4 \\ 1.65 \\ -2.7 \end{bmatrix} + \begin{bmatrix} 0.0 \\ 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 2.4 \\ 2.15 \\ -3.2 \end{bmatrix}$$

$$\mathbf{softmax}(\mathbf{z}) \cong \begin{bmatrix} 0.5611 \\ 0.4368 \\ 0.0021 \end{bmatrix}$$

  - 개 이미지 결과를 얻음.
- 연습: ϵ을 0.1일 때와 0.9일 때의 결과를 비교하기

FGSM Practice – PyTorch Tutorial

- Test model: [pytorch/examples/mnist]
- 현대적인 CNN 스타일의 간략한 예제 모델(이름만 LeNet입니다)

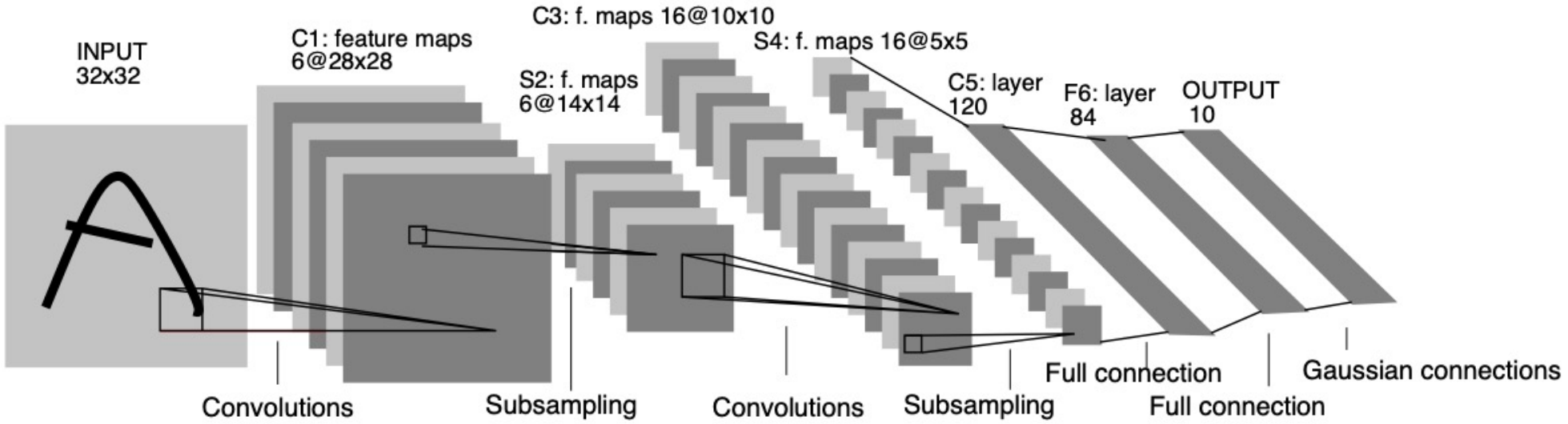


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Future Works

- NIPS 2017 적대적 공격과 방어에 대한 경쟁

  https://arxiv.org/pdf/1804.00097

- 음성-텍스트 변환 모델

  https://arxiv.org/pdf/1801.01944

- 배치, 병렬 공격

Introduction – Generative Adversarial Networks

- https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf
- 두 개의 서로 다른(distinct) 모델로 구성
  - 생성자(Generator): 학습한 이미지들과 같아 보이는 가짜 이미지 생성
  - 판별자 또는 구분자(Discriminator): 이미지를 보고 이것이 실제 학습 데이터에서 가져온 것인지, 또는 생성자에 의해 만들어진 가짜 이미지인지 판별
- 판별자가 항상 신뢰도 50%로 생성자의 출력이 진짜인지 가짜인지 판별할 수 있는 상태까지 두 모델이 경쟁

Introduction – Generative Adversarial Networks

- 손실 함수: $\min_G \max_D V(D,G) = \mathbb{E}_{x-p_{data}(x)}[\log D(x)] + \mathbb{E}_{z-p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$
- 판별자
  - x : 이미지
  - $D(x)$: 판별자의 신경망, x가 통과하면 확률 값을 출력함
  - 질문: 학습 데이터에서 가져온 x와 생성자가 만들어낸 x 중 $D(x)$는 뭐가 클까요?
- 생성자
  - z : 정규분포에서 뽑은 잠재 공간 벡터(laten vector space), 정규분포에서 n개의 원소를 추출해 구성한 벡터
  - $G(x)$: z 벡터를 원하는 데이터 차원으로 대응시키는 신경망
- 알고리즘
  - $D\left(G(z)\right)$: 실제 이미지 여부를 나타내는 0~1 사이의 확률 값(scalar)
  - D 는 이미지가 진짜인지 가짜인지 여부를 판별하는 확률인 $logD(x)$ 를 최대화하려고 함.
  - G 는 D가 가짜라고 판별할 확률인 $log\left(1 - D\left(G(z)\right)\right)$ 를 최소화하려고 함.
- 정답: $p_{data} = p_g$
  - p_{data} : 데이터셋 샘플을 뽑아 얻는 실제 데이터 분포(확률밀도함수)
    - 물리적 의미: 현실 세계의 손글씨가 분포한 공간 (예: 진짜 '3' 글자가 여러 스타일로 퍼져 있음.
  - p_g : $z{\sim}p_z(z)$ 정규분포로부터 샘플링한 z 를 생성자 G 에 넣어 $x = G(z)$ 를 생성
    - 처음엔 아무 의미 없는 이미지들 분포이며, 학습에 따라 점점 p_{data}를 닮아가는 이미지 분포

DCGAN

https://arxiv.org/pdf/1511.06434

- 생성자와 구분자에서 합성곱 신경망(convolution)과 전치 합성곱 신경망(convolution-transpose)을 사용

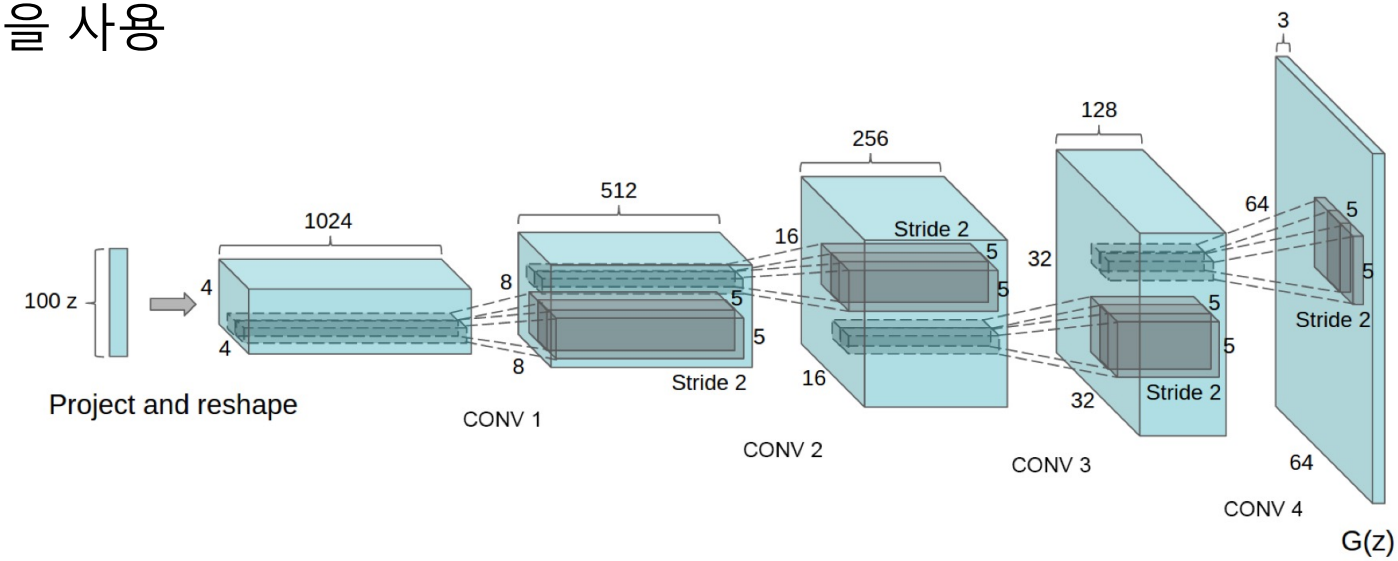


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN

- 이제 구현해 봅시다.