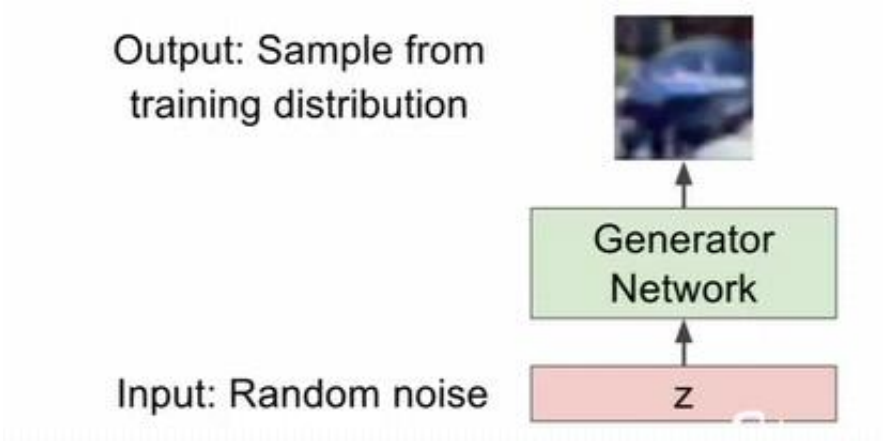
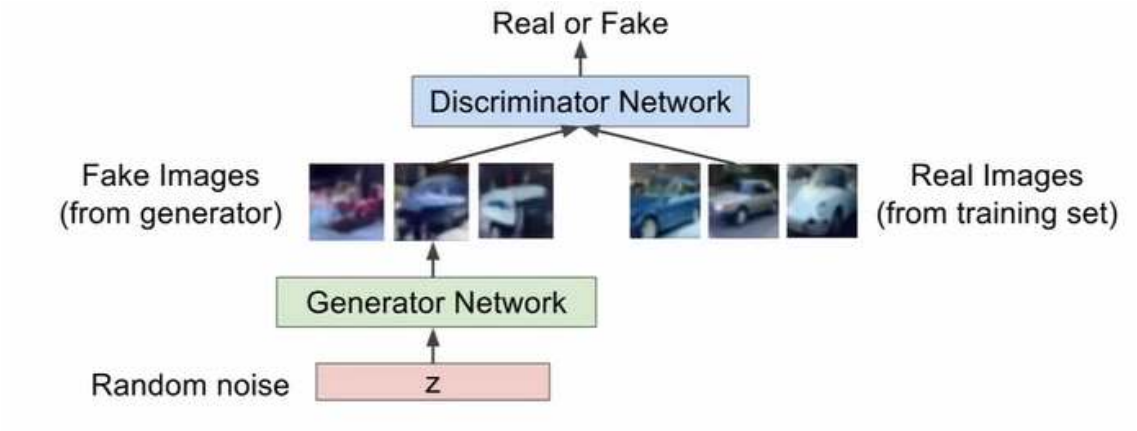


• gan 에서는 입력으로 random noise 벡터 z 를 받는다. (이 차원은 우리가 직접 명시해 준다.)
• 그리고 입력 z 가 생성 네트워크를 통과하면 학습 분포로부터 직접 샘플링 되어서 값을 출력한다.
• 즉 모든 random noise 입력이 학습 분포의 샘플에 매핑되길 원한다.
• gan 은 two player game 의 학습법을 이용한다. (Generator 과 Dscriminator)
• generator 은 사실적인 이미지를 형성해 discriminator 를 속이는게 목적
• discriminator 는 generator 가 만든 이미지가 가짜인지 아닌지를 잘 구분하고 싶어한다.



• random noise가 generator 의 입력으로 들어가고, generator 은 가짜 이미지를 생성한다.
• discriminator 은 가짜/실제를 구별 가능해야 한다.
• gan의 아이디어 : discriminator가 잘 학습해서 진짜 이미지인지 가짜 이미지인지를 잘 구분할 수 있게 되다면, generator 은 discriminator 을 속이기 위해 더 실제같은 가짜 이미지를 만들려 할 것 이다. 즉 서로가 학습을 도우면서 같이 성장.

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}})\right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

$\mathbb{E}_{x\sim p_{data}}\log D_{\theta_d}(x)$

- $\log D(x)$: 실제 데이터 x가 데이터 분포 p_d 에 속할 likelihood

$\mathbb{E}_{z\sim p(z)}\log(1-D_{\theta_d}(G_{\theta_g}(z)))$

- z 는 generator 에서 형성하고 (random noise)
- $G_{\theta_g}(z)$ 는 그에 따라 생성된 가짜이미지
- $D_{\theta_d}(G_{\theta_g}(z))$ 는 가짜이미지에 대한 Discriminator 의 출력

- 이 때 Discriminator 과 Generator 의 목적이 서로 다르다.
- Discrinimator 의 경우 실제 데이터와 가짜 데이터를 구분하고 싶은데에 목적이 있다.
- Generator 의 경우는 Discriminator 를 속이고 싶어한다.(즉 잘 구분 못하게 하고싶다)

[Discriminator 의 입장]

- Object function 을 최대화 하고싶다.

$\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}$

- real data에 대한 부분이 1에 가까울 수록 좋고

$-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}]$

- 가짜 데이터의 경우는 값이 0이 나올수록 좋다.
- (즉 '잘 구분' 하고 싶다는 것이다.)

- Generator 의 경우는 위와 반대이다.
- 즉 위의 object function 을 최소화 시키고 싶다.(disctiminator 를 속이고 싶다.) 즉 이는 진짜같은 이미지를 잘 만들어내고 싶다는 것이다.

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

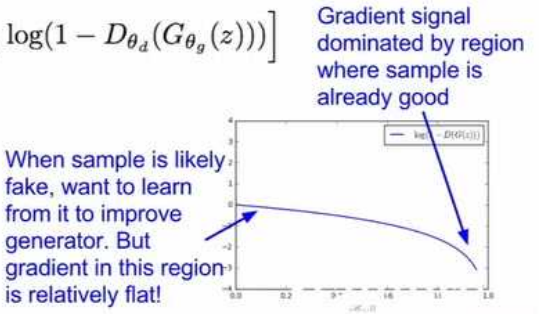
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

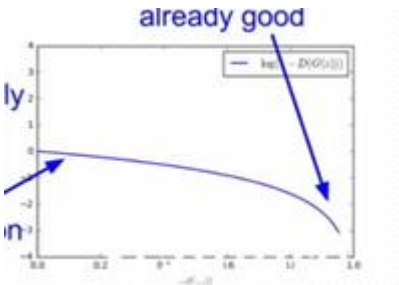
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!



• GAN을 학습시키려면 Generator 과 Discriminator를 번갈아가면서 학습시킨다.
• 이때에 discriminator 는 object를 maximize 하기 위해서 gradient ‘ascent’ 를 사용
• 반면에 generator 는 object 를 minimize 하기 위해서 gradient ‘descent’ 를 사용



• 이때에 중요한 사실이 있는데, 1-d(g(z)) 의 함수를 보면, 기울기가 오른쪽으로 갈수록 작아진다.
• 다시말해 generator가 이미지를 잘 생성할 때에 gradient가 매우 큰 반면(d(g(z)) 값이 대부분 1일 것이므로) generator 가 좋지 않을때는, gridient 가 매우 평평함을 볼 수 있다
• 즉 generator 가 ‘안좋을때’ 오히려 학습이 안된다. 이는 generator 를 학습시키는게 매우 어렵다는것이다.

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

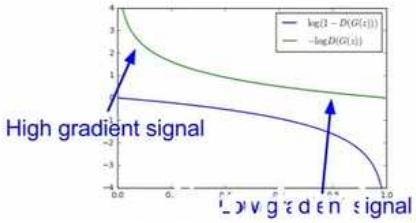
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



• 이를 해결하기 위해서 목적함수를 뒤집어서 보자. $\max \log D(G(z))$ 로 바꾸겠다는뜻.
• discriminator가 정답을 잘 맞출 likelihood 를 최소화 하는 방법 대신에 discriminator 가 틀릴 likelihood 를 최대화 시키는 쪽으로 학습시키자.
• 그러면 뒤집어진 object function 을 최대화 하려 하기 때문에 안좋은 sample 을 생성할 때의 gradient 가 매우 큼을 알 수 있다.
• 두 object function 모두 목적은 동일하지만 후자가 훨씬 좋다.
• 대부분의 gan 이 후자의 object function 을 써서 학습시킨다고 한다.

Putting it together: GAN training algorithm

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

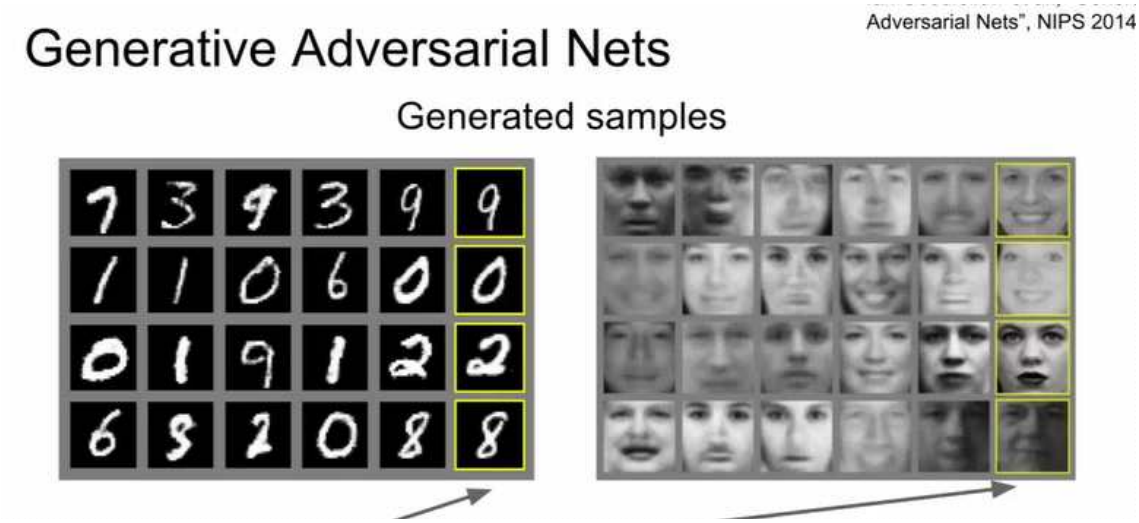
  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

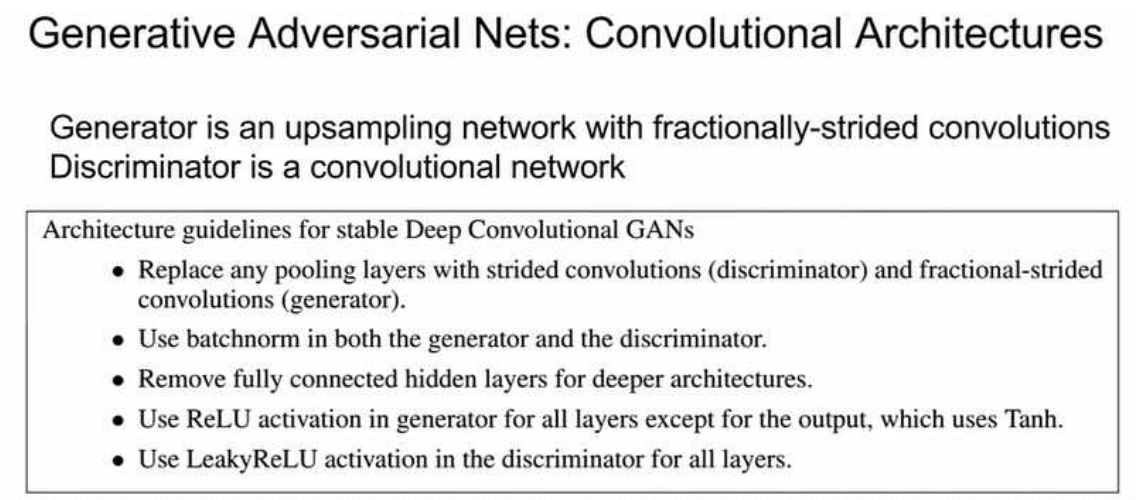
end for

SUMMARY
1. noise prior에서 미니배치만큼 노이즈를 샘플링
2. 그리고 학습데이터 x에서 실제샘플을 미니배치만큼 생성
3. 샘플링한 노이즈를 generator 에 통과시키면 가짜 이미지가 생성된다.
4. 그러면 미니배치만큼의 가짜,진짜 이미지를 얻는다.
5. discriminator 의 그라디언트를 계산시 위 이미지 set 을 이용하여 parameter 를 k step 만큼 학습시킨다.

6. 이제는 generator 를 학습시킨다.
7. noise prior 에서 노이즈를 샘플링
8. 샘플링된 노이즈를 generator에 통과시키고, generator 를 학습시킨다. (generator 는 가능한 discriminator 를 속이려고 하게 학습할 것이다.)
9. 위처럼 반복 (discriminator 과 discriminator)

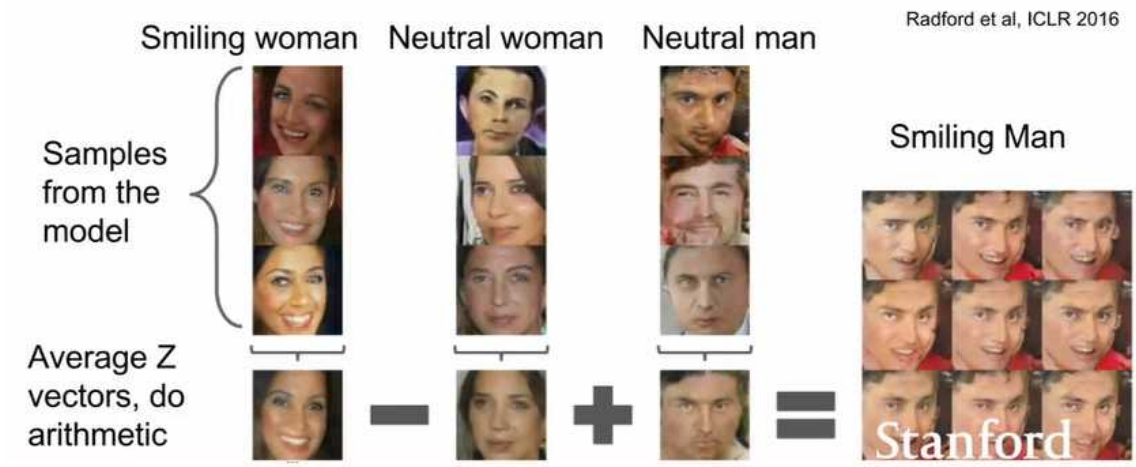


- 맨 오른쪽 이미지는 실제 이미지. 왼쪽은 가짜 이미지
- 이는 모델이 진짜같은 이미지를 잘 생성시키고 있음을 알 수 있다.

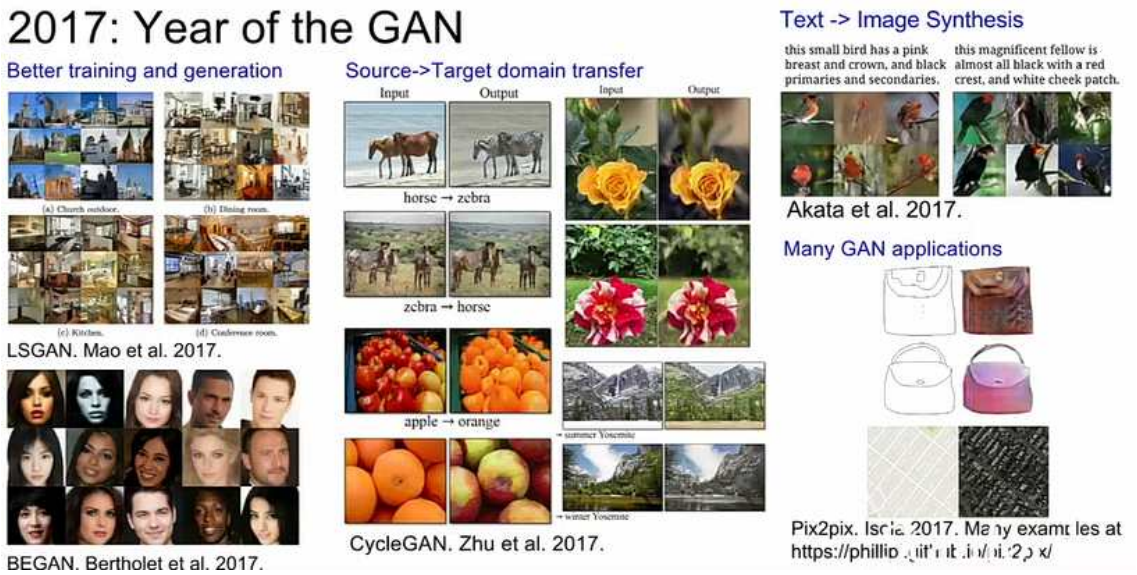


- A.Radford 가 ICLR16 에 발표한 CNN 구조를 GAN 에 적용한 연구가 GAN 의 성능을 극적으로 끌어올림 (DCGAN)

• 랜덤 벡터(z)의 두 방향을 변하게 하면서 생성한 이미지들. 매우 smooth 하게 변하고있음을 볼 수 있다.



• 벡터 z를 가지고 벡터연산을 해 볼 수 도 있다.
• z 를 더하고 / 빼는것을 통해서 이런 직관을 얻을수도 있다.
• 웃는 여자 - 여자 + 남자 = 웃는남자



• 좌측은 gan 을 잘 학습시키고 이미지를 생성하는 모델
• 중앙은 gan 으로 source -> target domain 변환도 가능하다. (말 -> 얼룩말, 사과 -> 오렌지....)
• 또는 이미지를 비싼 카메라로 찍은듯한 느낌으로 변환해줄 수 도 있다.
• 겨울에 촬영한 이미지를 여름에 촬영한 이미지로 해줄 수 도 있다.
• 텍스트로 된 설명을 입력으로 받아서 이미지를 생성할 수 도 있다.
• 색을 칠해주는것도 가능하다.

GANs

Don't work with an explicit density function
Take game-theoretic approach: learn to generate from training distribution through 2-player game

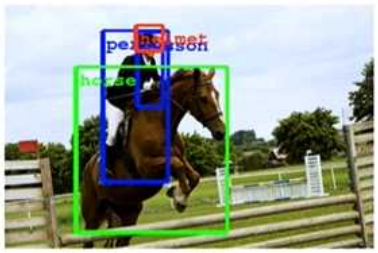
Pros:
- Beautiful, state-of-the-art samples!

Cons:
- Trickier / more unstable to train
- Can't solve inference queries such as p(x), p(z|x)

Active areas of research:
- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

• Gan 은 특정한 확률분포를 정의하지 않는다.
• 대신에 샘플을 활용한 게임이론을 이용하였다.
• 데이터의 퀄러티가 매우 좋다.
• 학습이 까다롭고, 불안정하다. (object dunction 에 두개의 모델이 간섭)
• p(x) , p(z|x) 를 구할 수 없다.
• 많은 연구가 진행중인 분야이다.

Since 2013, deep neural networks have matched human performance at...



(Szegedy et al, 2014)

...recognizing objects and faces....

(Taigmen et al, 2013)

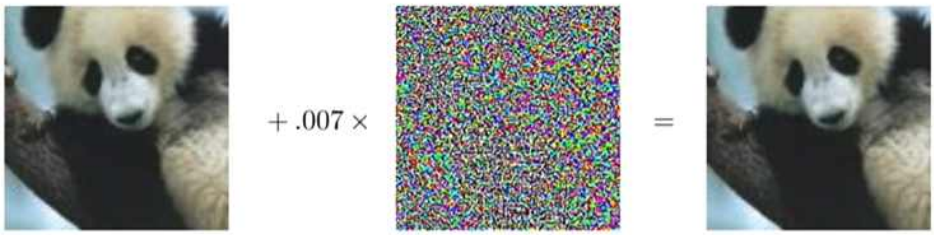
(Goodfellow et al, 2013)

...solving CAPTCHAS and reading addresses...

(Goodfellow et al, 2013)

- 2013 년 이후로 딥러닝은 사람의 인식능력을 뛰어넘고 있다.
- 사람 얼굴인식, 글자 인식 (더이상 CAPTCHAS 를 사용하지 않는 이유이다.) 은 큰 발전
- 하지만 컴퓨터도 실수를 저지른다.

Adversarial Examples



Timeline:
"Adversarial Classification" Dalvi et al 2004: fool spam filter
"Evasion Attacks Against Machine Learning at Test Time"
Biggio 2013: fool neural nets
Szegedy et al 2013: fool ImageNet classifiers imperceptibly
Goodfellow et al 2014: cheap, closed form attack

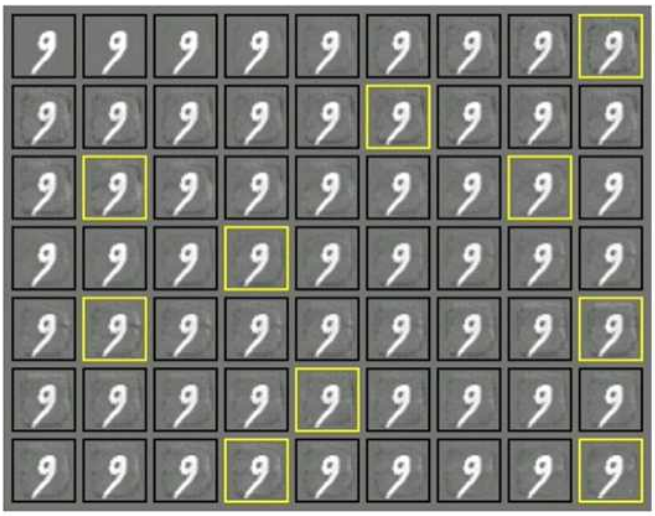
(Goodfellow 2016)

- 왼쪽은 판다. CNN 는 당연히 99%로 판다라고 판단.
- 오른쪽 판다는 60% 로 이게 판다라고 판단한다.
- 즉 이미지에 약간의 회손(노이즈 추가) 만으로 CNN 는 실수를 하기 쉬워진다.

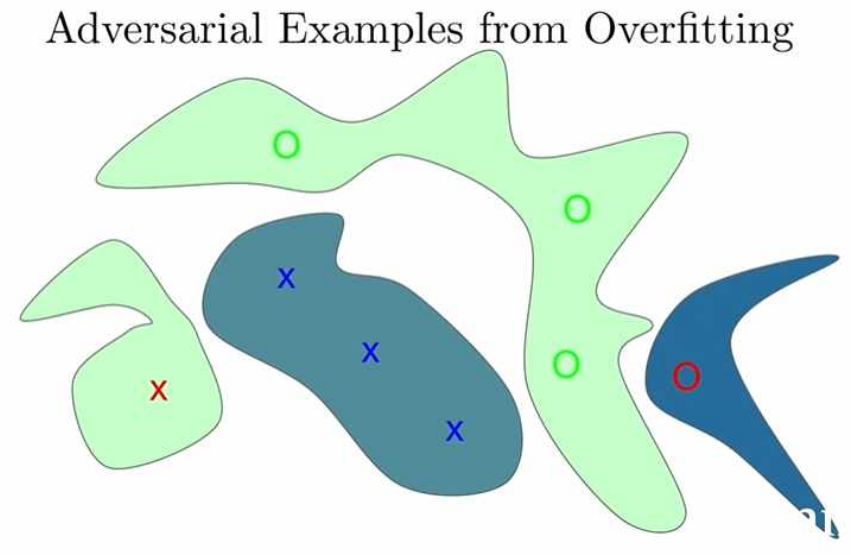
Turning Objects into “Airplanes”

• 위 경우 왼쪽 상단부터 오른쪽 하단까지의 transform 을 통해 cnn 이 비행기라고 인식하게된다.

Attacking a Linear Model

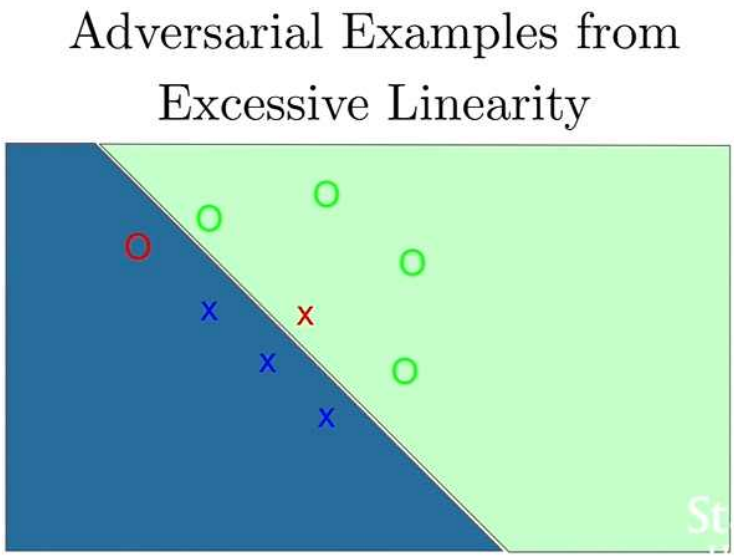


• 이는 복잡한 cnn 의 경우만 해당되지 않는다. 매우 간단한 머신러닝도 속기 쉽다.
• linear model(매트릭스를 곱하고, bias 더하고 soft max를 적용한 단순한 모델)
• 왼쪽 상단 -> 오른쪽 하단 으로 갈수록 모델이 0으로 판단하는 비율이 높아진다.
• 노란 박스는 모델이 0 이라고 0.9 이상 확신을 가지고 생각하게된 큰 점



- x, o 를 구분하게 모델을 fitting 하게 했다고 하자.
- 파란색 x , 녹색 o 는 모델이 train 시 사용한것들
- 녹색영역 : 모델이 o 로 판단, 파란색영역 : 모델이 x 로 판단
- 복잡한 function 이라 녹색 지역이 다른곳에서도 나타난다. (복잡한 형태로)
- x 의 근처에서 X를 만들어낸다고 해도, O 로 판단하는 지점에 형성이 되어서 O 라고 오인할 수 있다.

사실 이런 실수는 다른 간단한 모델에서 만들어진다.



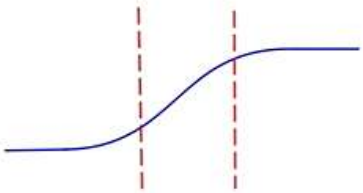
- 같은 경우이다. 모델이 간단해도, 살짝만 변하는경우에 잘못판단하고 있다.

Modern deep nets are very piecewise linear

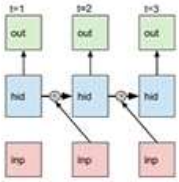
Rectified linear unit

Maxout

Carefully tuned sigmoid

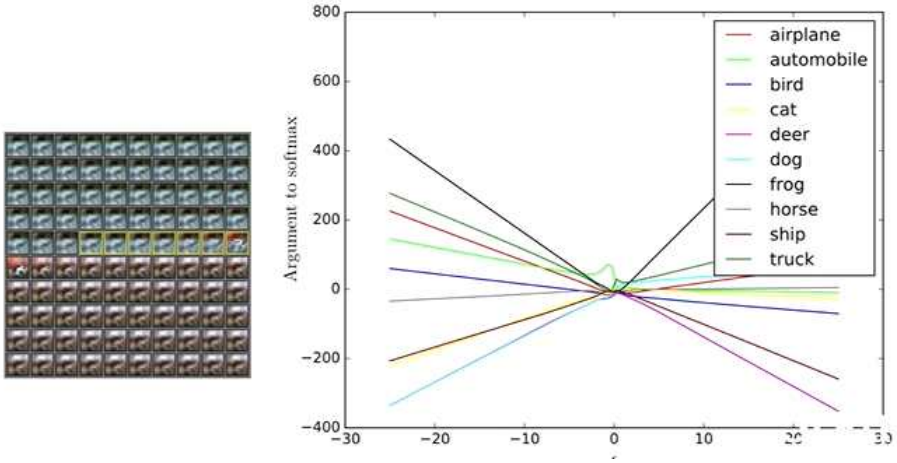


LSTM



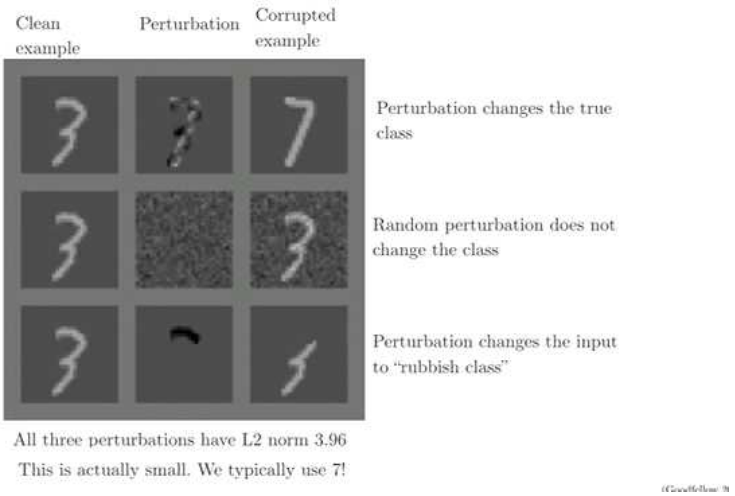
- 요즘 딥러닝 모델은 pieswise linear 하다고 한다.
- relu 는 linear
- maxout 는 pieswise linear
- sigmoid (가운데 지점은 거의 linear)
- LSTM(시간이 지나면서 정보를 잃지 않으려 단순한 + (linear) 통로가 있다. 시간이 지나갈수록 linearity 가 심해짐)
- 지금 말하는 이 linearlity 는 바로 input 과 ouput 관계에서의 linearlity
- 하지만 paramter 과 output 과의 관계는 nonlinear (왜냐하면 weight matrix at each layer of matrix 는 계속 서로 곱해지므로, 즉 non linear reaction 이 점점 커진다.)
- 이런 현상들이 training NN, optimizing paramter을 어렵게 만든다.

Nearly Linear Responses in Practice



- input space 에서 one-dim traverse 의 변형을 생각해보자.
- 원본 이미지는 하얀차 + 빨간 배경
- direction 을 정한다. 이 direction 에 곱할 epsilon 을 정의
- epsilon 이 -30 (왼쪽 상단) , 0이면 original image
- epsilond 이 -30 ~ 30 으로 변하는 과정을 나타낸다.
- 노란색은 input에 대해 올바르게 '차' 로 모델이 예측.
- 오른쪽은 logit 값을 나타낸다. 변형이 심할수록 frog 로 인식한다는것을 볼 수 있다.
- cnn이 매우 복잡한 모델이라, 확률의 추이가 복잡할거라 생각할것같지만 output 은 linear 처럼 생겼다.

Small inter-class distances



- 모든 변화는 같은 L2 NORM PETUBATION 을 가진다.
- 맨 위의 줄을 보면 3 -> 7 , 두번째줄은 노이즈 추가
- 작은 individual pixels의 changes 는 모두 더해지면 큰 값이 된다.
- 그림이 크면 each pixel 의 아주 작은 변화가 엄청난 변화를 누적하게됨

The Fast Gradient Sign Method

$$J(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \approx J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x}).$$

Maximize

$$J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

$$||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_\infty \leq \epsilon$$

$$\Rightarrow \tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})\right).$$

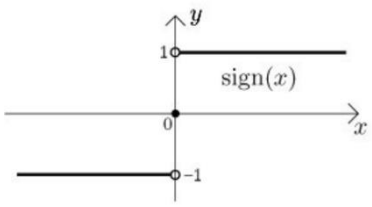
.

- 개념
  - FGSM은 신경망의 그래디언트(gradient)를 이용해 적대적 샘플을 생성하는 빠르고 간단한 기법이다. 입력 이미지에 대한 손실 함수의 그래디언트를 계산하여 그 손실을 최대화하는 이미지를 생성하는 것이다. 이처럼 새롭게 생성된 이미지가 적대적 이미지(adversarial image)

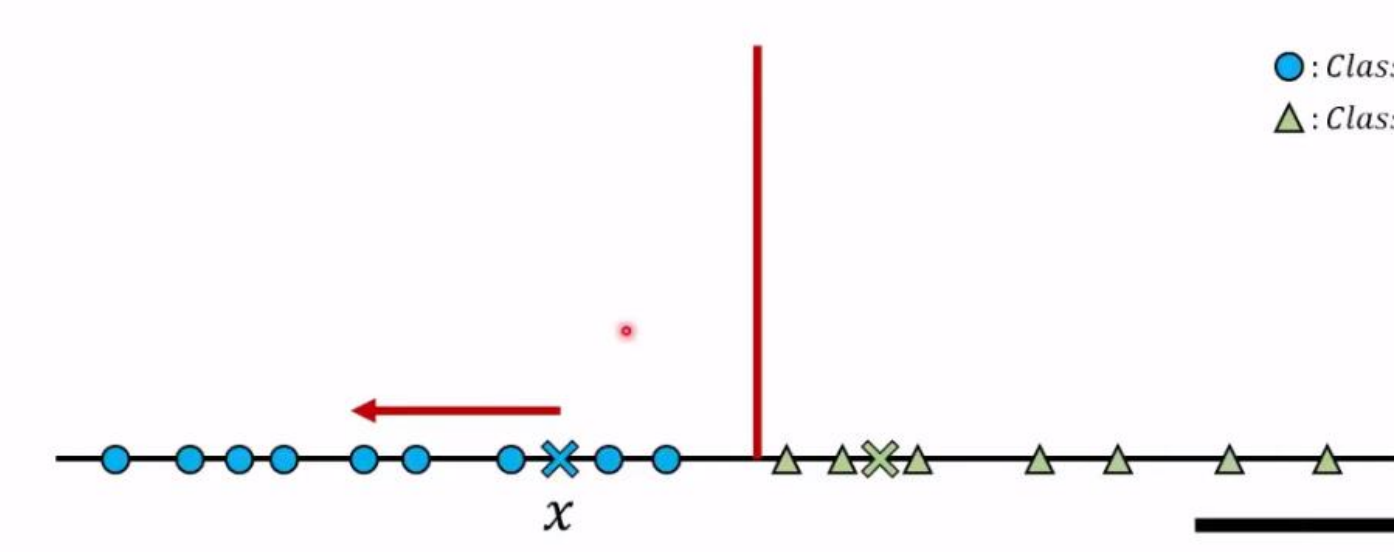
- 변수설명
  - $\tilde{x}$: 적대적 이미지, x : 원본 입력 이미지.
  - ϵ : 각 픽셀이 얼마나 변형 가능한지 나타내는 상수(왜곡의 양을 적게 만들기 위해 곱해지는 작은 값)
  - θ : 모델의 파라미터, J : 손실 함수

$$\text{sign}(x) = \begin{cases} 1, & x > 0; \\ 0, & x = 0; \\ -1, & x < 0. \end{cases}$$

  - 원래는 cost의 기울기와 반대방향으로 움직여 cost를 0으로 만드는 데 오히려 cost gradient의 부호를 그대로 받아 cost를 크게 하는 방향으로 update

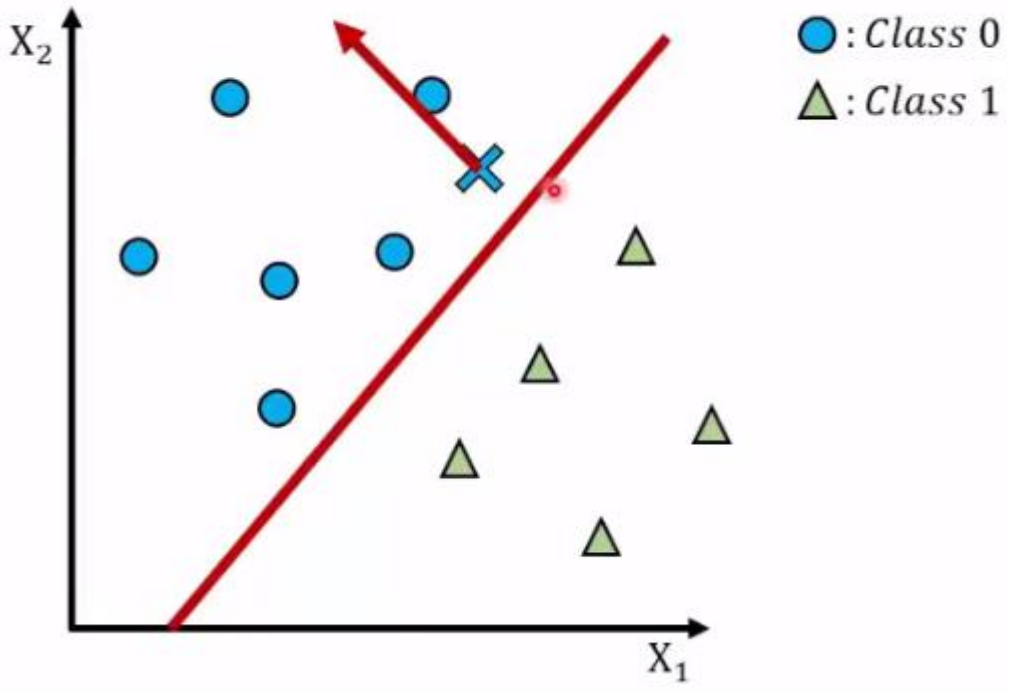


- **FGSM** 원리
  - 1차원

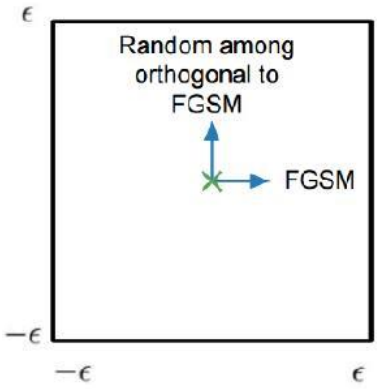


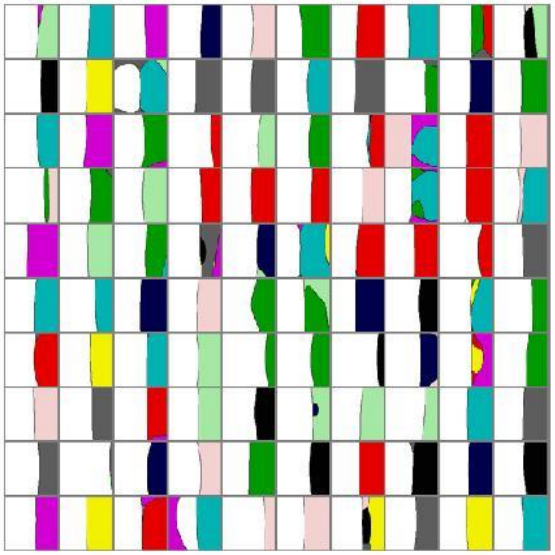
  - X가 더 확실히 '동그라미 Class' 로 분류되기 위해서 cost를 줄이는 방향인 왼쪽방향으로 움직여야 함
  - X를 그 반대방향인 오른쪽으로 ϵ 만큼 보내면 decision boundary 를 넘어 '세모Class' 로 분류됨

  - 2차원



Maps of Adversarial and Random Cross-Sections



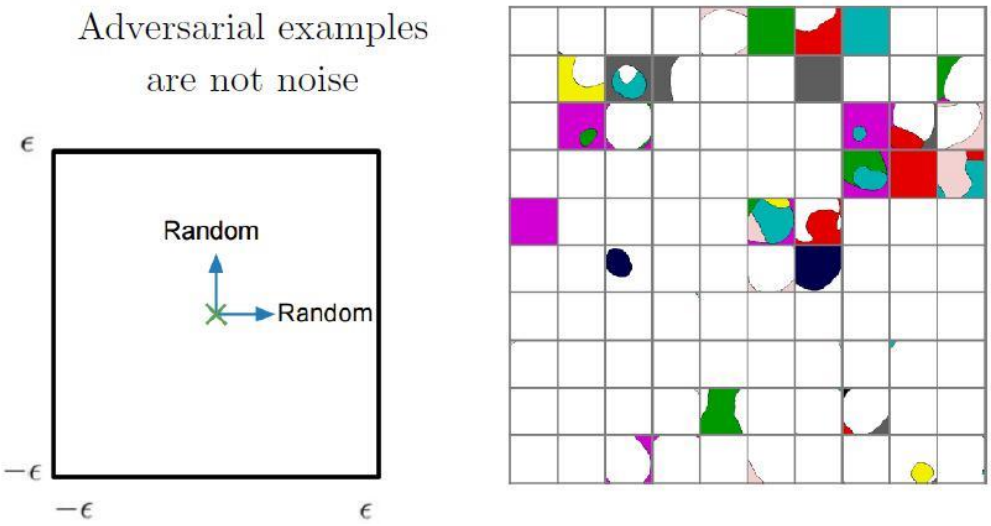


- 인공신경망 decision boundary의 사상
  - 2차원 이미지의 각 데이터가 어떻게 분류되어야 했는지를 나타내는 것이다. 각각의 작은 사각형은 CIFAR-10분류기의 결정경계의 매핑이다,

- 각 셀의 작용원리
  - 셀의 중심은 아무 변형없는 본래 샘플에 해당한다. 왼쪽과 오른쪽은 FGSM의 공격방향을 나타내고 위 아래 방향은 공격방향에 수직인 임의의 방향이다.
  - 흰 픽셀들은 올바른 분류, 다른 색의 픽셀들은 다른 카테고리를 의미함
  - 대부분의 셀에서 왼쪽은 흰색 오른쪽은 다른 색을 띄고 그 경계선이 선형인 모습 확인가능
    - FGSM이 방향을 인식한다는 의미이고 큰 내적을 얻는 방향으로 적대적이미지를 생성하면 됨

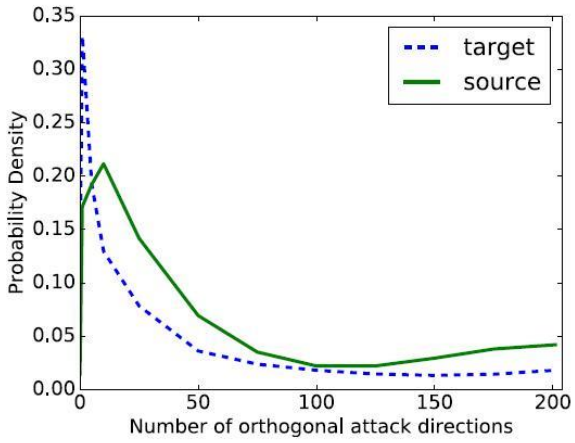
- 실제로 각 샘플은 선형의 decision boundary에 가까이 위치하고 있고 그 경계만 넘기면 그 주변은 다른 class 의 샘플들이 존재 . 따라서 실제로 올바른 방향만 알고있으면 정확한 공간좌표를 찾을 필요 없이 적대적 이미지 생성하여 분류기를 속일 수 있음

Maps of Random Cross-Sections



- 적대적 이미지는 노이즈가 아니다.
    - 두 축 모두 랜덤하게 노이즈를 생성하면 올바르게 분류된 이미지는 여전히 올바르게 분류되고 적대적 이미지도 그대로이다.
    - 몇몇 잘못 분류된 부분은 test set 이기 때문, 애초에 잘못 분류된 이미지에 노이즈를 더해도 결과는 같다.

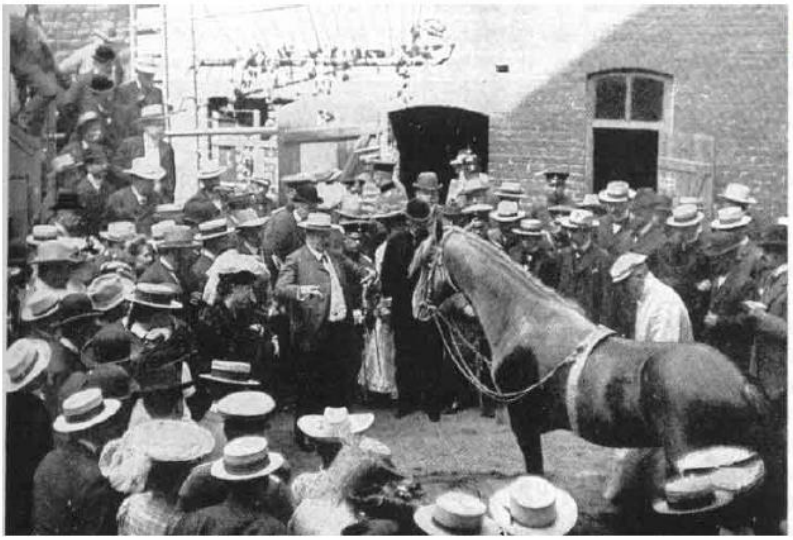
Estimating the Subspace Dimensionality



(Tramèr et al, 2017)

- 적대적 이미지가 놓여있는 적대적 영역(adversarial region / adversarial subspace)의 차원을 알아내면, 노이즈생성을 통해 적대적 이미지를 만들 수 있을지 알 수 있다. (모든 방향에서 적대적 이미지를 생성한다면 어떤 변화든 분류오류를 일으킴)
- 2차원에서 적대적 영역은 평균 25차원이다.
- 적대적 영역의 차원은 적대적 이미지의 전이와 밀접한 관련이 있음. 두 개의 다른 모델이 있을 때 적대적영역이 클수록 적대적 이미지가 전이될 가능성이 높다. 반대로 적대적 영역이 작다면 두 모델이 완벽히 동일한 부분공간을 지니지 않는 이상 전이되기 어렵다.

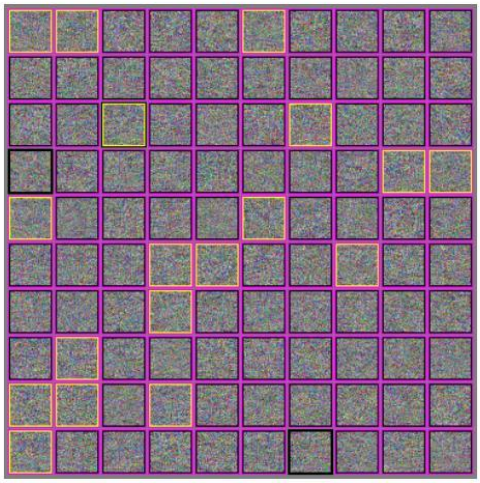
Clever Hans



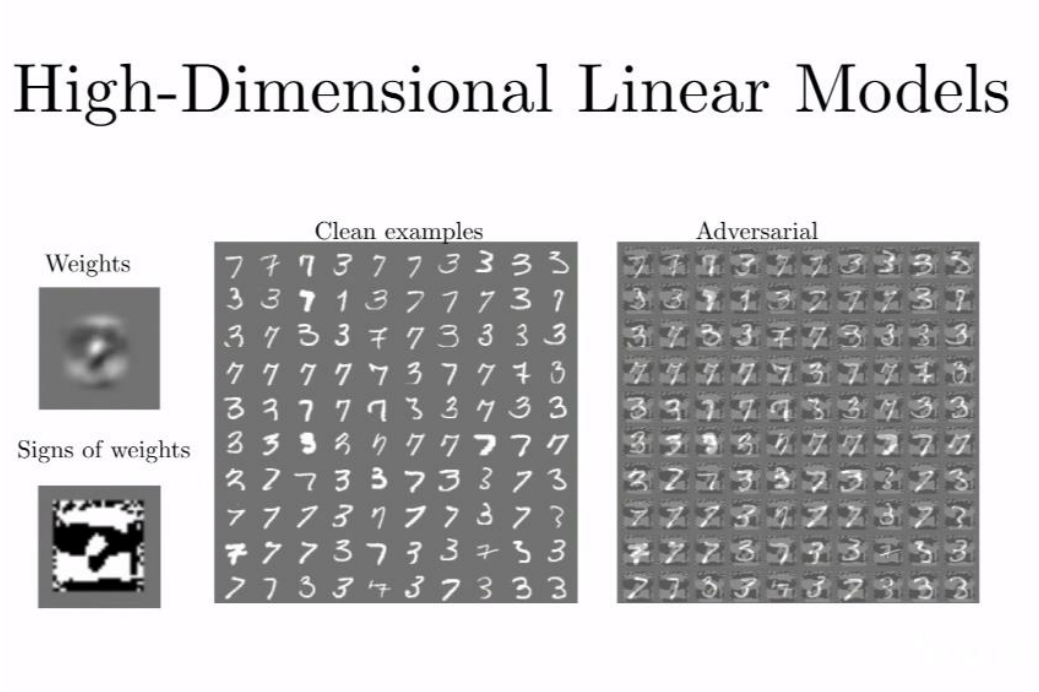
("Clever Hans, Clever Algorithms," Bob Sturm)

- 머신러닝 알고리즘에 대한 비유
  - 1900년대 초반, 한스라는 말이 주인의 "2+1이 뭐지?"와 같은 산수질문에 발을 굴러 대답함.
  - 하지만 마스크 쓴 사람과 단 둘이 진행하니까 맞추지 못함. 실제로 한스가 산수를 하는 줄 알았으나 알고보니 사람들의 사회적인 반응을 통해 정답을 맞추게 된 것. 결과를 일반화하지 못함
  - 머신러닝도 이와 유사하게 training data와 유사한 분포를 찾아내고 test data에 일반화시키지만 고의적으로 모델을 속이기 위한 샘플하나로 모델을 쉽게 속일 수 있는 것이다.
  - NN , deep RL 모델도 잘 속음

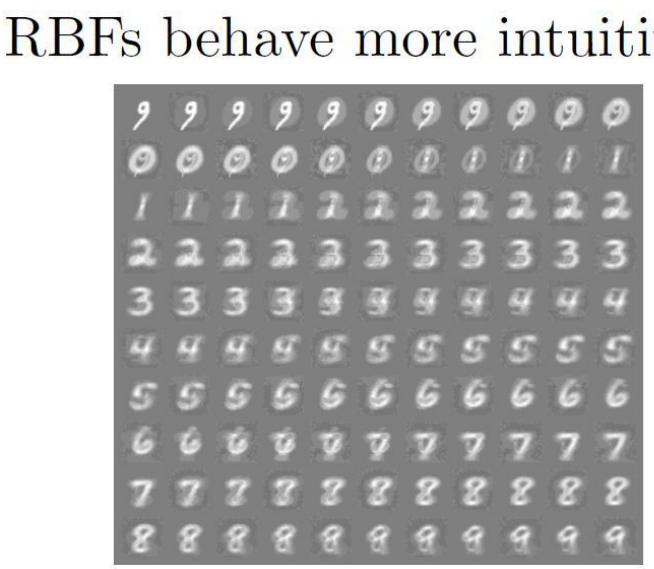
Wrong almost everywhere



  - 현대의 머신러닝 알고리즘은 대부분 틀림. IID의 test set sample 로 데이터를 대체하면 거의 모든 샘플 잘못 분류된 것을 알 수 있을 것이고, 모델은 train data 근처에서만 좋은 성능을 발휘함

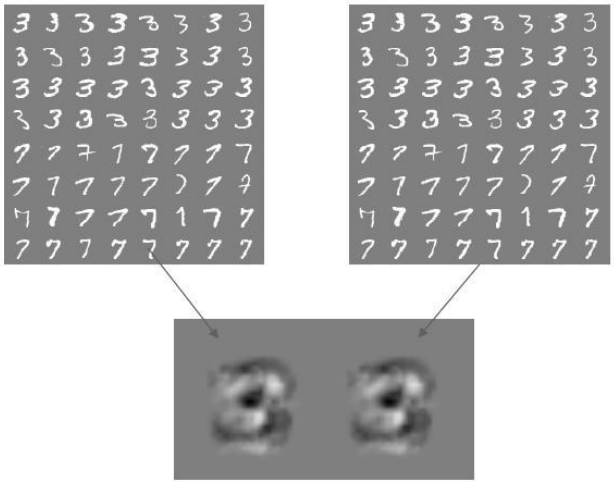


- 3과 7을 구별하기 위한 로지스틱 회귀모델
  - 3과 7의 차이의 평균을 시각화한 가중치와 가중치의 sign을 얻을 수 있음
  - 이를 이미지에 더하거나 빼면 사람에겐 의미 없는 attack 이 모델입장에서는 가장 중요한 지표로 작용하여 속음



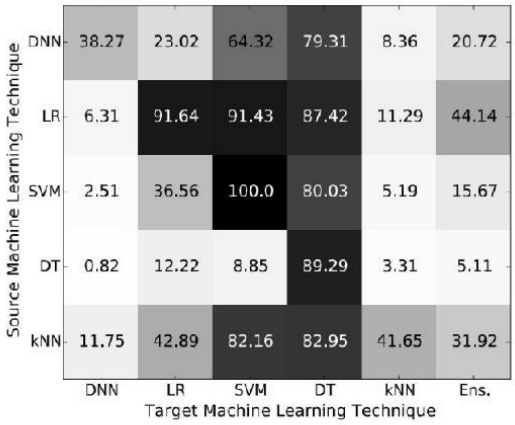
- 잘 작동하는 선형모델
  - 얕은 RBF(Radial Basis Function 방사형네트워크) 가 FGSM공격에 강함
  - 문제는 RBF는 그레디언트가 거의 모든 층에서 소실되기 때문에 층을 깊게 쌓기 어렵고 이로인해 정확도가 낮다는 점이다.
  - Batch normalization 기법 써도 어려움
  - 더 좋은 hyper parameter 혹은 최적화 알고리즘으로 해결할 수 있을 것으로 보임

Cross-model, cross-dataset generalization



- 데이터 집합에서 다른 데이터 집합으로, 한 모델에서 다른 모델로의 일반화
  - 두 개의 다른 training data로 두 개의 다른 모델을 학습시킨다. 두 training set 은 MNIST 3과 7을 분류하는 작은 데이터. 왼편에 있는 데이터와 오른편에 있는 데이터를 각각 학습시켜 아래의 가중치를 얻음.
  - 두 가중치 벡터는 굉장히 유사한 모습을 보이는데 이는 머신러닝 알고리즘이 일반화 되었기 때문.
  - 학습시키고자 하는 objective function은 training data와 무관함. 즉 어떤 training 샘플을 선택하는지는 중요하지 않다. test set을 training set으로 일반화하려 할 때도 마찬가지
  - 각 모델이 유사한 함수를 익혔기 때문에 비슷한 적대적 이미지에 취약함

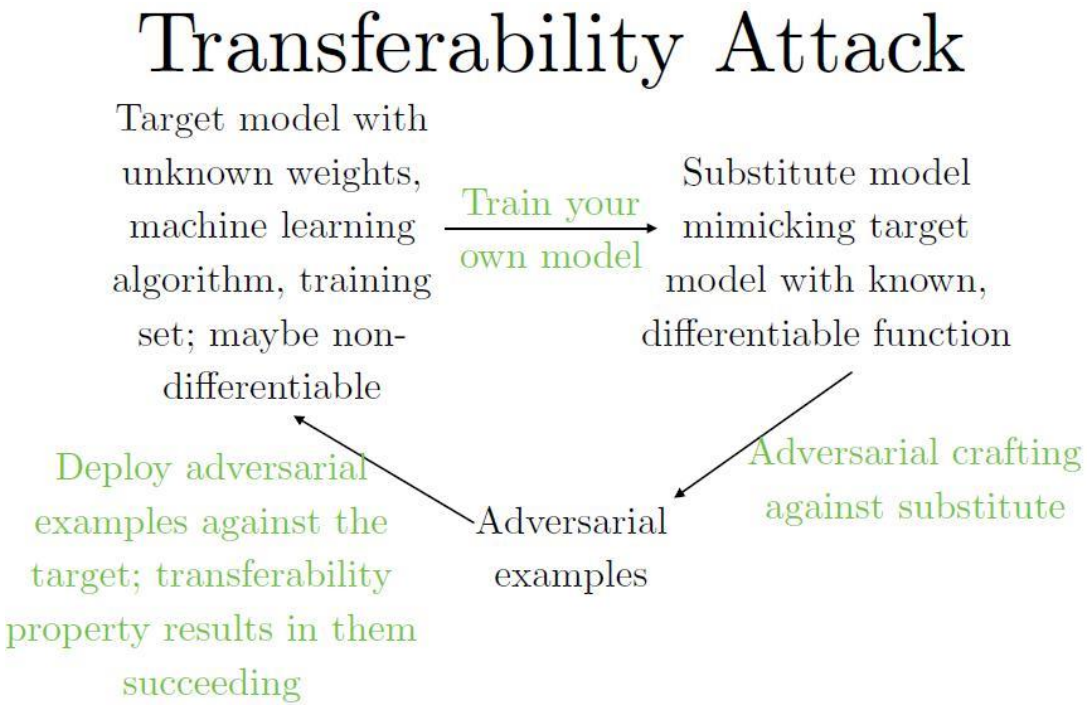
Cross-technique transferability



(Papernot 2016)

- 머신러닝 기법들에 대한 전이가능성
  - 서로 다른 데이터셋 사이에서 전이될 뿐만 아니라 여러 머신러닝 기법들 간의 전이도 가능

- A기법에서 작동한 적대적 이미지가 B기법에서 작동하는 비율
- Ex) Logistic Regression 에서 Decision Tree 로 전이되는 비율이 87.42%
- 검정색 박스 - 높은 전이율



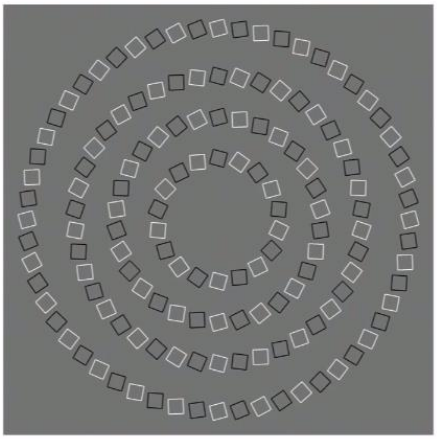
  - Unknown 모델에게 공격하는 방법
    - 가중치, 알고리즘, 데이터 셋, 파라미터 모두 모름
    - 방법1. 직접 labeling 한 이미지로 인식기 만듦으로써 target 모델과 '같은 적대적 이미지' 공유하게끔
    - 방법2. Unknown 모델에 입력이미지 넣고 출력값을 받아 이를 train set으로 활용

Enhancing Transfer With Ensembles

	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
-ResNet-152	17.17	0%	0%	**0%**	0%	0%
-ResNet-101	17.25	0%	1%	**0%**	0%	0%
-ResNet-50	17.25	0%	0%	2%	0%	0%
-VGG-16	17.80	0%	0%	**0%**	6%	0%
-GoogLeNet	17.41	0%	0%	**0%**	0%	5%

- 앙상블을 통한 전이율 증가
  - 모델을 앙상블하고 적대적 이미지 적용했더니 0%에 가까운 낮은 정확도 보임(모두 속임)

Adversarial Examples in the Human Brain



These are concentric circles, not intertwined spirals.

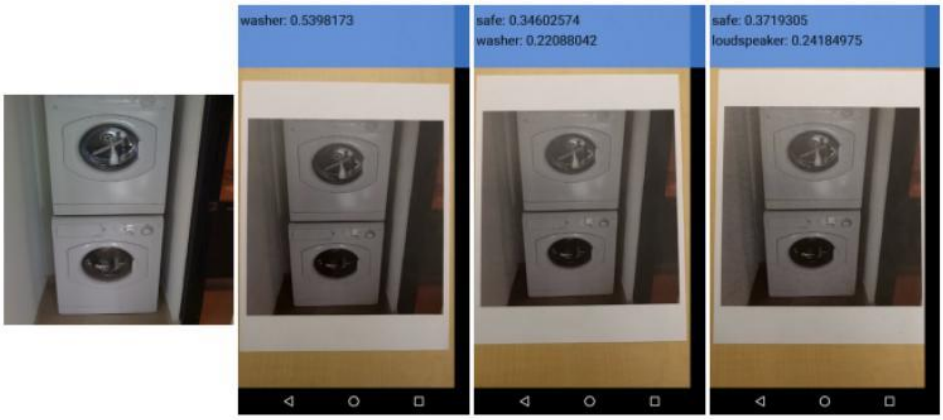
(Pinna and Gregory, 2002)

(Goodfello

Practical Attacks

- Fool real classifiers trained by remotely hosted API (MetaMind, Amazon, Google)
- Fool malware detector networks
- Display adversarial examples in the physical world and fool machine learning systems that perceive them through a camera
  - Nicolas Papernot : 적대적 샘플 전이를 통해 구글, 아마존 등의 분류기를 속이는 데 성공함
    - 각 기업의 API에 data 업로드 해서 train data 구축하고 model 카피하는 형식

Adversarial Examples in the Physical World



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

(Kurakin et al, 2016)

- 카메라에 적대적 이미지 보여줌
- 카메라상의 인식기와 적대적 이미지를 생성하는 모델은 서로 다름(분리됨)
- 따라서 attacker가 모델이나 interface에 직접적으로 접근하지 못하더라도 약간의 물리적인 변형으로 시스템을 속일 수 있는 것

Failed defenses

Generative pretraining

Removing perturbation with an autoencoder

Adding noise at test time

Ensembles

Confidence-reducing perturbation at test time

Error correcting codes

Multiple glimpses

Weight decay

Double backprop

Adding noise at train time

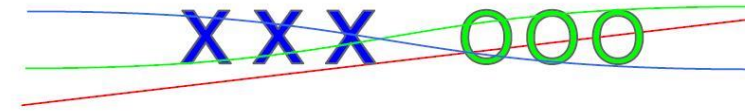
Various non-linear units

Dropout

- 위의 10가지 공격 방법에 대한 방어책을 구축하는데 실패함
  - Traditional regularization으로는 해결 불가

Universal approximator theorem

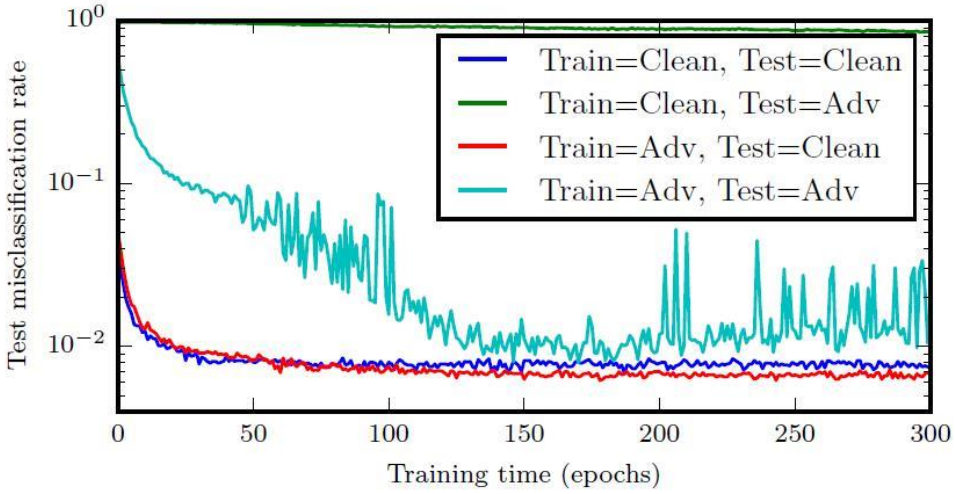
Neural nets can represent either function:



Maximum likelihood doesn't cause them to learn the right function. But we can fix that...

- Universal Approximation Theorem
  - 1개의 히든 레이어를 가진 Neural Network를 이용해 어떠한 함수든 근사시킬 수 있다는 이론

Training on Adversarial Examples



- 적대적 샘플로 학습을 한다면?
  - 원본데이터로 학습하고 적대적 샘플로 테스트 : 거의 다 속임
  - 적대적 샘플로 학습과 테스트 : 오류가 눈에 띄게 줄지는 않음
  - 적대적 샘플로 학습시키고 원본으로 테스트 : regularization 효과로 원본으로 학습했을 때보다 더 높은 성능

Adversarial Training of other Models

- Linear models: SVM / linear regression cannot learn a step function, so adversarial training is less useful, very similar to weight decay
- k-NN: adversarial training is prone to overfitting.
- Takeway: neural nets can actually become more secure than other models. *Adversarially trained neural nets have the best empirical success rate on adversarial examples of any machine learning model.*
  - 선형 모델 : SVM / 선형회귀모델은 step function 학습할 수 없으므로 adversarial training 효과 떨어짐
  - K-NN : adversarial training 이 과적합 되기 쉬움
  - 경험적인 성공률로 보아 Neural Net 이 최선

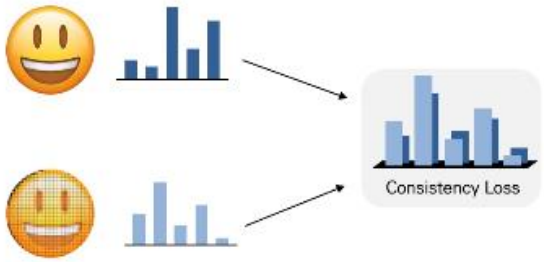
Weaknesses Persist

- Adversarial Training 의 문제
  - 모델을 구성할 때 서로 다른 카테고리에 대해 입력이미지를 최적화하기 어려움
  - Ex) CIFAR 10의 트럭을 나머지 각기 9개의 다른 카테고리로 변환하려 함. 이때 트럭과 유사하여 적합할 수 잇는 카테고리가 bird 카테고리 오직 하나만 존재함

- VAT(Virtual Adversarial Training)
  - Adversarial training 에 필수적인 label이 없다면?

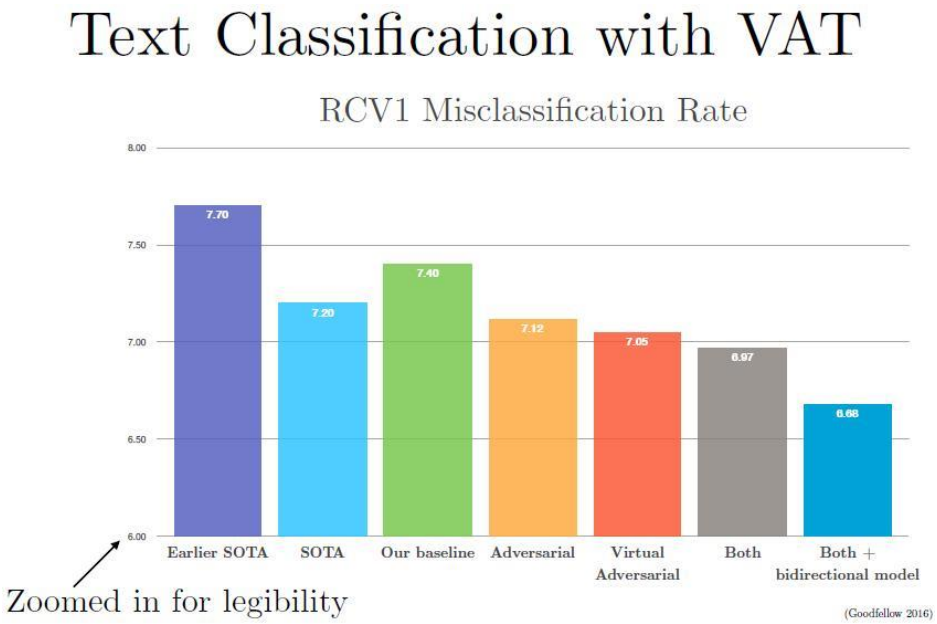
    1. Image에 최대한 noise를 줘서 Adversarial Image 생성
    2. Original Image와 Adversarial Image의 분포가 유사하도록 학습(Consistency Training)

Maximize KL-Divergence

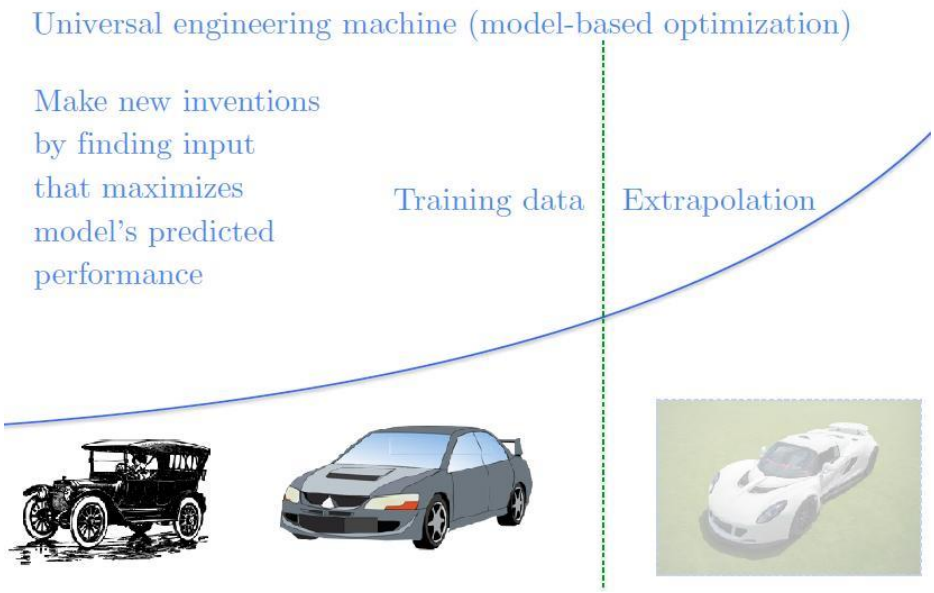


Minimize KL-Divergence

-
  - 적대적 이미지 생성할 때 동일한 label 유지



- VAT를 이용해 labeled data와 unlabeled데이터를 함께 학습하는 semi-supervised learning 을 구현할 수 있음. Text 에는 unlabeled data 많아 이를 해결한모델의 정확도 향상



- 모델 기반 최적화 문제
  - 엄청 빠른 슈퍼카를 개발한다고 하자
  - NN이 신차의 설계도를 보고 그 설계도를 통해 얼마나 빠른 차를 만들 수 있는 예측하게함
  - Input을 최적화하고 슈퍼카의 설계도를 모색. (실제로 없음)
  - 적대적 샘플을 생성해 모델이 이 차가 굉장히 빠르다고 생각하게 함
  - 이 적대적 샘플 문제를 해결하면 모델 기반 최적화를 구현할 수 있다.

  - 자동차, GPU, 신약개발 등 다양한 분야에 adversarial 기법을 적용할 수 있음

Conclusion

- Attacking is easy
- Defending is difficult
- Adversarial training provides regularization and semi-supervised learning
- The out-of-domain input problem is a bottleneck for model-based optimization generally

Reference

https://www.programmersought.com/article/339084548/

http://dsba.korea.ac.kr/seminar/?mod=document&uid=68