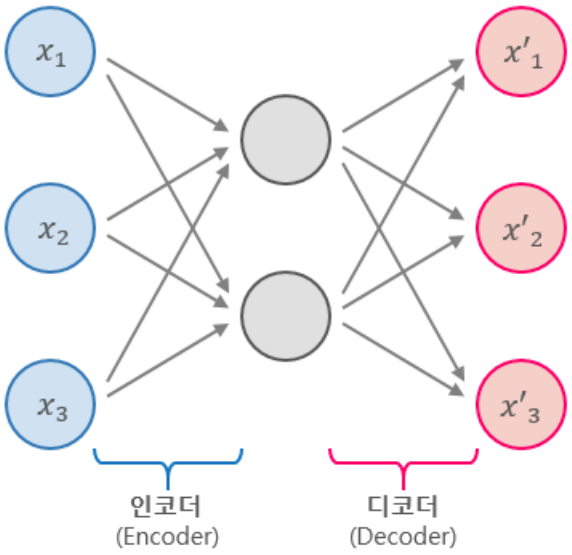
CH17 : Representation learning and Generative learning using Autoencoder and GAN

Jan 08, 2021

DGIST 융복합대학 기초학부 한현영(hyhan@dgist.ac.kr)

Autoencoder

- **Autoencoder**(이하 AE) : 어떤 지도 없이(labeling 되어 있지 않은 훈련 데이터를 사용) 잠재 표현 또는 코딩(일반적인 부호화)이라 칭하는 입력 데이터의 밀집 표현을 학습할 수 있는 인공 신경망 / 단순히 입력을 출력으로 복사하는 방법을 배움
- 생성 모델 : AE가 훈련 데이터와 매우 비슷한 새로운 데이터를 생성하는 것
- **GAN**(generative adversarial networks, 생성적 적대 신경망) : 신경망 두 개로 구성.
  - 생성자 : 훈련 데이터와 비슷하게 보이는 데이터를 생성
  - 판별자 : 가짜 데이터와 진짜 데이터를 구별
- encoder(인지 네트워크, recognition network) : 입력을 내부 표현으로 변환
- decoder(생성 네트워크, generative network) : 내부 표현을 출력으로 변환



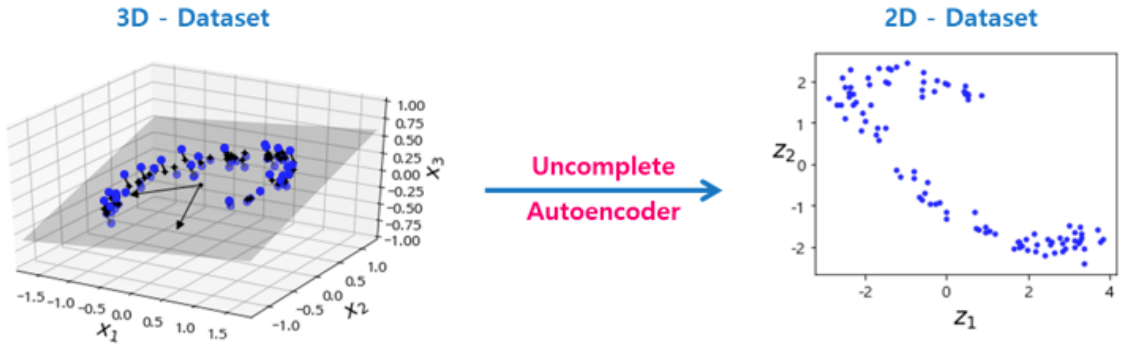
input과 output의 개수가 동일하며 대칭으로 구축

MLP와 유사한 모습(출력층의 뉴런 수가 입력 개수와 동일하다는 것 제외)

- AE가 입력을 재구성하기 때문에 출력을 재구성, reconstruction이라 함
- 비용함수는 재구성이 입력과 다를 때 모델에 벌점을 부과하는 재구성 손실, reconstruction loss 포함
- 내부의 표현이 입력 데이터보다 저차원(위의 예시의 경우 3 --> 2)이기 때문에 undercomplete autoencoder, 과소완전 오토인코더라 함

Undercomplete AE

AE가 선형 활성화 함수만 사용하고 비용 함수가 평균 제곱 오차(MSE)인 경우 PCA를 수행하는 것으로 간주 가능

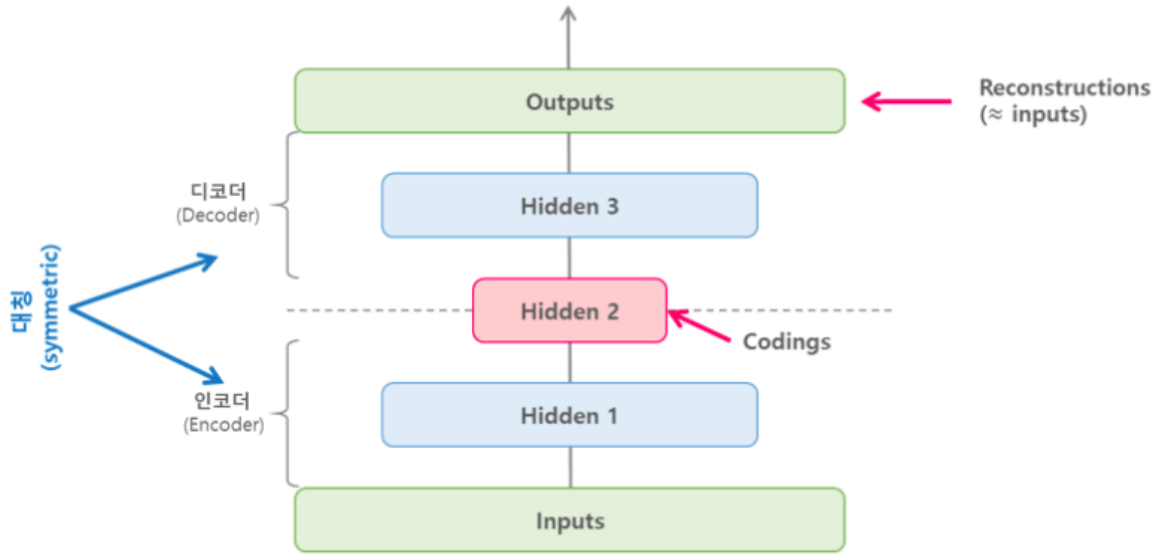


원본 3D dataset(left) & autoencoder의 hidden layer 출력(i.e. coding layer / right)

- AE는 데이터의 분산이 가능한 많이 보존되도록 데이터를 projection할 이상적인 2D 평면 탐색 (PCA와 같은...)

Stacked AE(Deep AE)

- 여러 개의 hidden layer를 가지는 AE, layer를 추가할수록 AE가 더 복잡한 코딩(부호화)을 학습 가능
- 가운데 hidden layer(coding layer)을 기준으로 대칭인 구조를 가짐



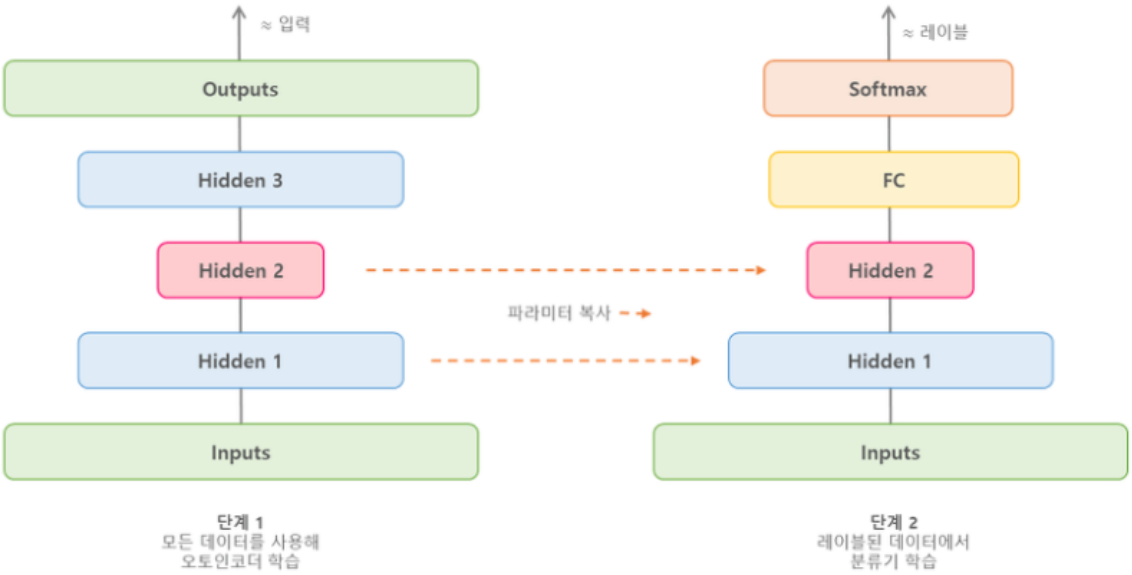
AE 훈련 척도 --> 입출력 비교

e.g. 재구성된 이미지가 정보를 잃는 경우 --> 모델을 더 오래 훈련하거나 인코더와 디코더의 층을 늘리거나 코딩의 크기를 늘릴 수 있음. 하지만 네트워크가 너무 강력하면 데이터에서 유익한 패턴 학습 X, 완벽한 재구성 이미지 생성 가능성 있음(과대적합)

[11장]
[https://github.com/h0han/2021_UGRP/blob/master/seminar/presentation/Chapter%2011%20%EC%8B%AC%EC%B8%B5%20%EC%8B%A0%EA%B2%BD%EB%A7%9D%20%ED%9B%88%EB%A0%A8%ED%95%98%EA%B8%B0.pdf]에서 언급한 바와 같이 레이블된 훈련 데이터가 많지 않은 지도 학습 문제를 다루어야 한다면, 비슷한 문제를 학습한 신경망의 하위층을 사용할 수 있음

유사하게 대부분 레이블되지 않은 대량의 데이터셋이 있다면 먼저 **전체 데이터를 사용해 Stacked AE를 훈련**, 그 다음에 AE의 하위층을 재사용해 문제를 해결하기 위한 nn을 만들고 레이블된 데이터를 사용해 훈련

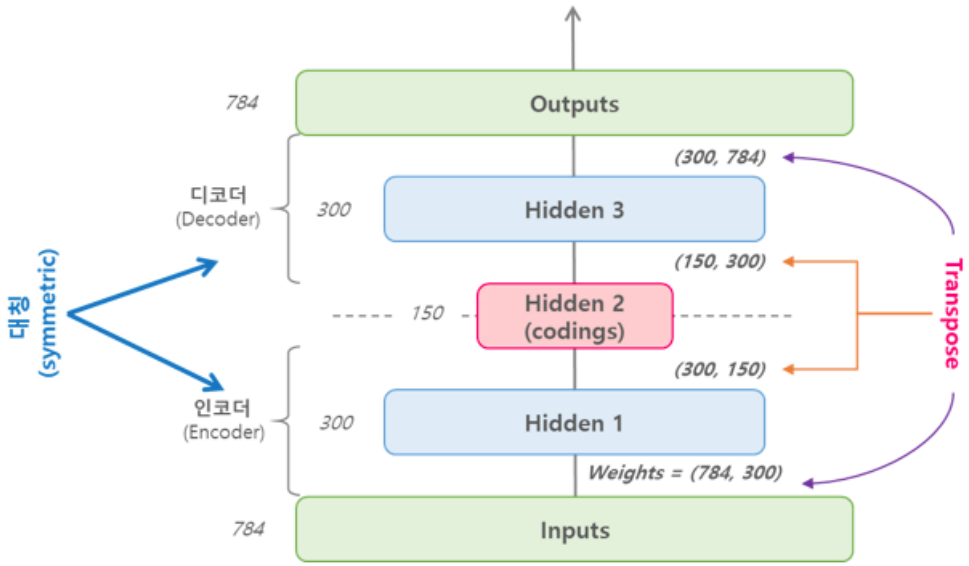
분류기를 훈련할 때 레이블된 훈련 데이터가 많지 않으면 사전 훈련된 층을 동결하는 것이 좋음



AE를 사용한 비지도 사전 훈련

가중치 묶기

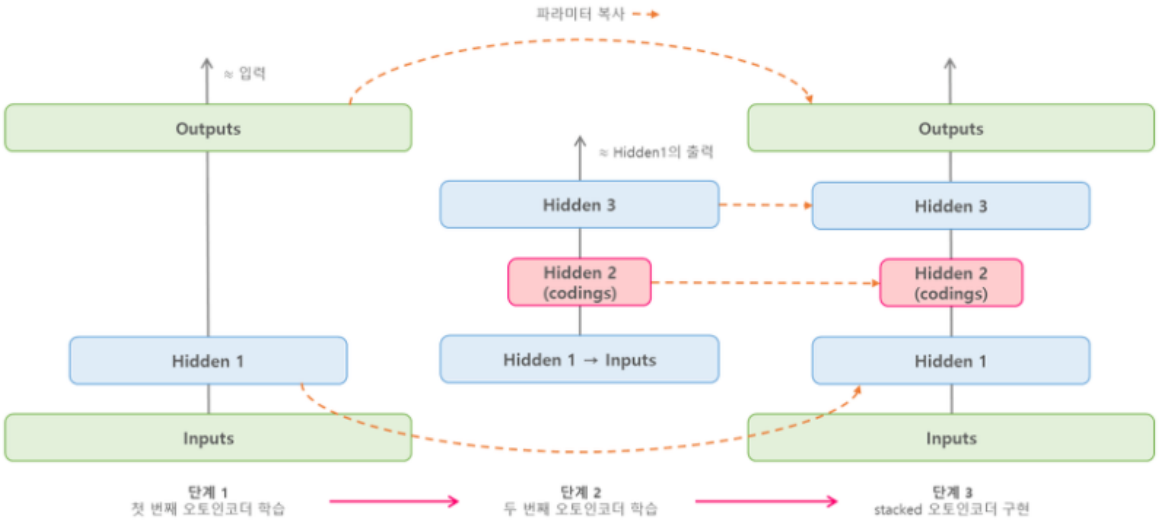
AE가 완벽하게 대칭일 경우 일반적으로 인코더의 가중치와 디코더의 가중치를 묶어줌 --> 네트워크의 가중치 수가 절반으로 줄어들어 **학습 속도를 높이고 overfitting의 위험을 덜어낼 수 있음**



한 번에 한 층씩 학습

아주 깊은 AE의 경우 한 번에 AE 하나를 학습하고, 이를 쌓아 올려 한 개의 stacked AE를 만드는 것이 유용

1. 첫 번째 AE는 input을 재구성하도록 학습
2. 1단계의 AE 이용, 압축된 new train set 생성. new train set에서 두 번째 AE 훈련
3. 모든 AE 이용하여 전체 network 구축



Convolutional AE

- 이미지를 다루는 경우

인코더는 Conv layer과 Pooling layer로 구성된 일반적인 CNN

- 인코더는 전형적으로 입력에서 공간 방향의 차원(높이와 너비)을 줄이고 깊이(특성 맵의 개수)를 늘림
- 디코더는 이미지의 스케일을 늘리고 깊이를 원본차원으로 되돌림

--> [전치 합성곱 층][https://kionkim.github.io/2018/06/08/Convolution_arithmetic/] 사용

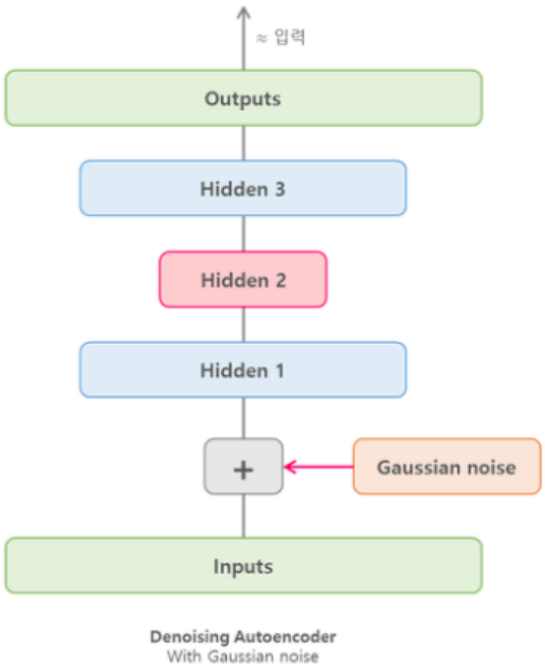
Recurrent AE

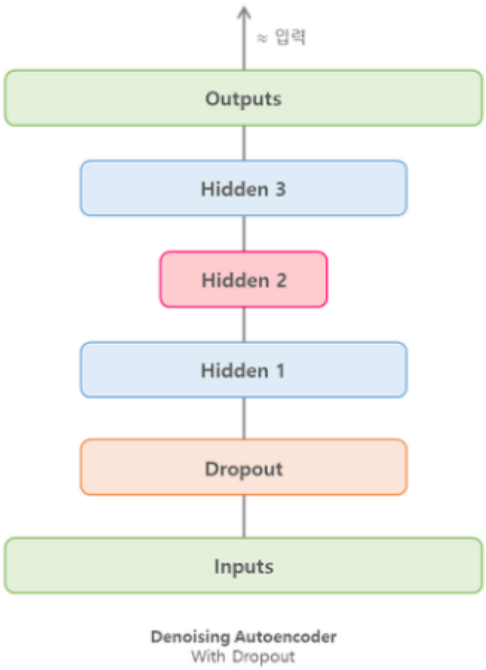
- 비지도 학습이나 차원 축소를 위해 시계열이나 텍스트와 같은 시퀀스에 대한 AE를 만들기 위해 순환 신경망(CH15 참고)이 밀집 네트워크보다 나을 수 있음
- 인코더는 **seq2vec** RNN
- 디코더는 **vec2seq** RNN

입력 크기 또는 입력보다 큰 coding layer를 두는 과대완전(overcomplete) AE 또한 생성 가능

Stacked denoising AE

AE가 의미 있는 feature을 학습하도록 제약을 주기 위해 input에 noise 추가하고, noise가 없는 원본 입력을 재구성하도록 학습 --> `Gaussian noise; 정규분포를 가지는 잡음, 일반적인 잡음`를 추가하거나 `dropout` 이용



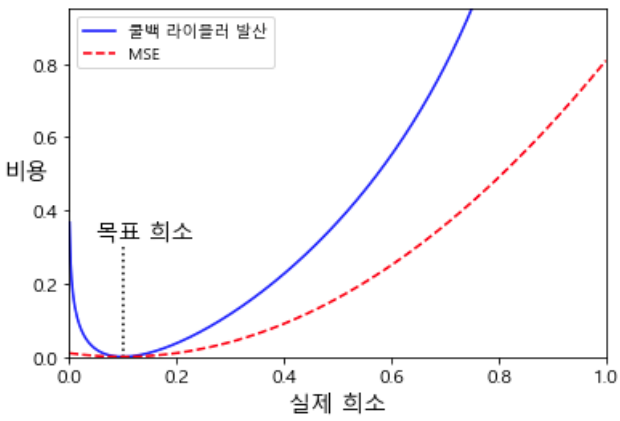


Sparse AE

좋은 특성을 추출하도록 만드는 다른 제약의 방식 --> 희소성(sparsity) 이용

- 비용 함수에 적절한 항을 추가하여 **AE가 coding layer에서 활성화되는 뉴런 수를 감소하도록** 함
  - 가령 coding layer에서 평균적으로 5% 뉴런만 활성화 되도록 만들어 주게 되면, AE는 5%의 뉴런을 조합하여 입력을 재구성해야 하기 때문에 유용한 특성을 표현하게 됨
- 훈련 반복마다 coding layer의 **실제 희소 정도를 측정하고 측정된 희소 정도가 타깃 희소 정도와 다르면 모델에 벌칙 부과**
  - 전체 훈련 배치에 대해 coding layer에 있는 각 뉴런의 평균적인 활성화 계산
- 목표 희소보다 활성화되는 경우 규제 필요 `1. 제곱 오차 추가`, `2. 쿨백-라이블러 발산 사용`

> Kullback-Leibler divergence : 평균 제곱 오차보다 훨씬 강한 gradient를 가짐



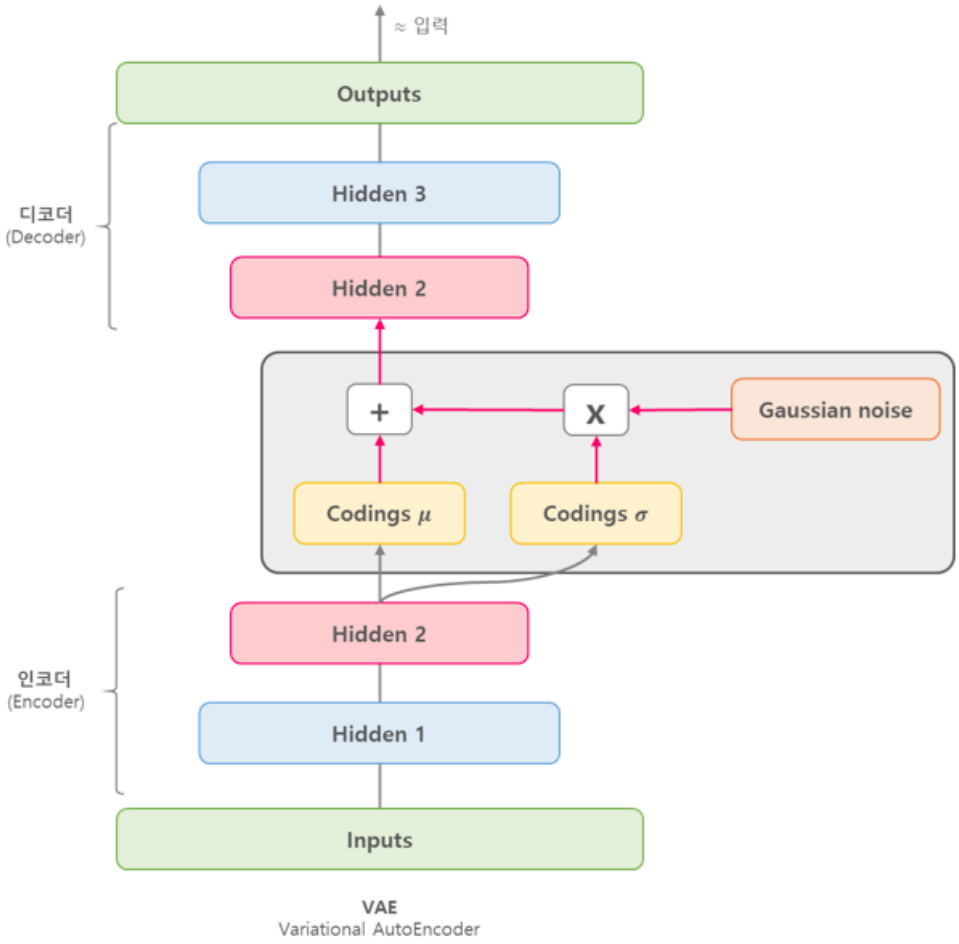
coding layer의 각 뉴런에 대해 희소 손실을 구하고 이 손실을 모두 합한 뒤 비용 함수의 결과에 더함. 희소 손실과 재구성 손실의 **상대적 중요도를 제어**하기 위해 희소 손실에 희소 가중치 하이퍼파라미터를 곱함

- 가중치가 너무 크면 모델이 목표 희소에 가깝게 되겠지만 입력을 적절히 재구성하지 못해 쓸모없는 모델이 될 수 있음
- 가중치가 너무 작으면 모델이 희소 목표를 거의 무시할 것이므로 어떤 흥미로운 특성도 학습하지 못함

Variational AE (VAE)

GAN에 비해 학습이 안정적인 편; 손실함수에서 확인할 수 있듯 *reconstruction error*과 같이 평가 기준이 명확하기 때문

- 확률적 AE(probabilistic AE); 학습이 끝난 후에도 출력이 부분적으로 우연에 의해 결정
- 생성적 AE(generative AE); 훈련 세트에서 샘플링된 것 같은 새로운 샘플 생성 가능



VAE의 coding layer은 다른 AE와 달리 주어진 입력에 대해 바로 coding을 생성해내는 것이 아니라 인코더는 **평균 코딩 mu와 표준편차 코딩 sigma**를 만듦

실제 coding은 평균이 mu이고 표준편차가 sigma인 가우시안 분포에서 랜덤하게 sampling되며, 이렇게 sampling된 coding을 디코더가 원본 입력으로 재구성하게 됨

- VAE는 마치 가우시안 분포에서 샘플링된 것처럼 보이는 코딩을 만드는 경향이 있는데, 학습하는 동안 손실함수가 코딩(coding)을 가우시안 샘플들의 집합처럼 보이는 형태를 가진 코딩 공간(coding space) (또는 **잠재 변수 공간**(latent space))로 이동시키기 때문

이러한 이유로 VAE는 학습이 끝난 후에 새로운 샘플을 가우시안 분포로부터 랜덤한 코딩을 샘플링해 디코딩해서 생성할 수 있음

Loss function

1. AE가 입력을 재구성하도록 만드는 일반적인 **재구성 손실**(reconstruction loss)
2. 가우시안 분포에서 샘플된 것 같은 코딩을 가지도록 AE를 제어하는 **잠재 손실**(latent loss)

추가적으로 목표 분포(가우시안 분포)와 실제 코딩 분포 사이의 KL 발산 사용

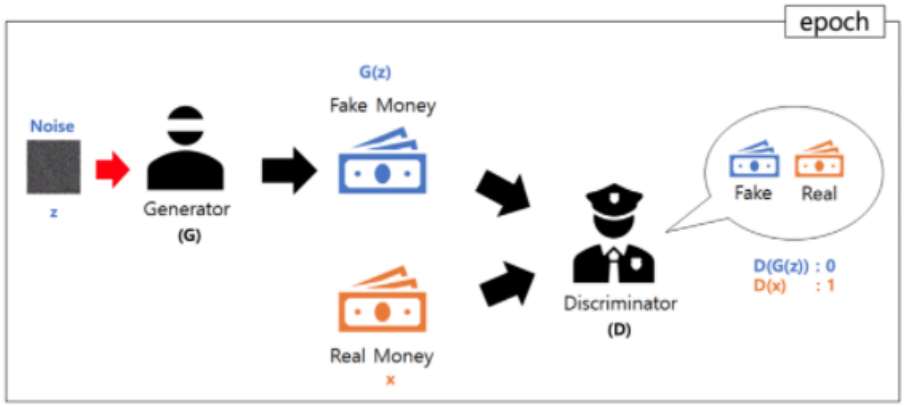
reference : https://ratsgo.github.io/generative%20model/2018/01/27/VAE/

semantic interpolation

VAE는 픽셀 수준의 보간 대신 코딩 수준에서 두 이미지 보간 가능

Generative Adversarial Network, GAN 😎

위조 지폐범이 위조 지폐를 만들면 경찰은 해당 지폐가 진짜인지 아닌지를 구별하게 되고 이를 반복하다 보면 위조 지폐범이 점차 더욱 진짜 같은 위조 지폐를 만듦



1. 제일 처음 0과 1로 인해 마구 뿌려진 Noise라는 쓰레기가 있다. 해당 Noise는 위 수식에서 z라고 표현된다.
2. 해당 쓰레기를 가지고 Generator(위조지폐범이) 위조지폐를 만든다. G가 z를 가지고 만들었으니 만들어진 위조지폐들을 위 수식에서 G(z)라고 하자.
3. 이제 Discriminator(경찰) 이 위조지폐와 실제 지폐를 구분해야 한다. 경찰은 이게 위조지폐라면 0을 출력하고, 진짜 지폐라면 1을 출력하기로 한다. 위조지폐 G(z) 와 실제 지폐 x 가 경찰 손으로 들어갔을 때, D(G(z))는 위조지폐이기 때문에 0, 실제 지폐는 D(x)는 1을 출력하게 된다.

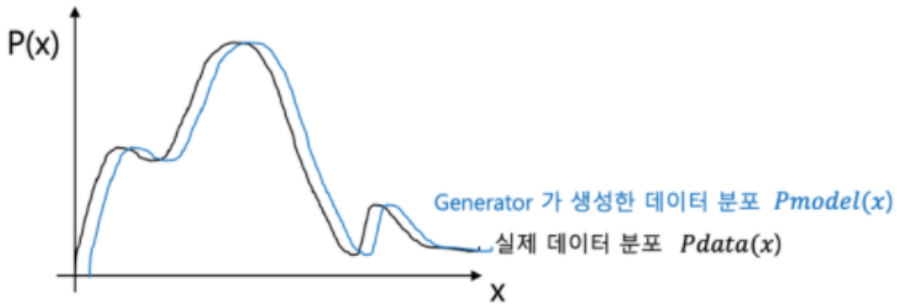
이렇게 한번 경찰이 해당 지폐들을 구분한 게 한 번의 epoch 가 된다. 첫 번째 도전에서 경찰이 위조지폐와 진짜 지폐를 잘 구분할지라도, 점차 위조 지폐 범은 더욱 비슷하게 생긴 위조지폐를 만들려고 노력할 것이고 경찰도 점차 위조지폐를 더 잘 구분하기 위해 노력할 것이다.

그러다가 어느 순간 너무나도 완벽한 위조지폐가 탄생한다면, 경찰은 결국 해당 지폐를 구분하지 못하기 때문에 이게 진짜인지 가짜인지 찍기 시작할 것이다. 확률은 둘 중 하나일 테니 결국 50%로 가 될 것이고, 그 순간 학습이 끝나게 된다.

Generator

Goal : 판별자를 속일 만큼 진짜 같은 이미지 생성

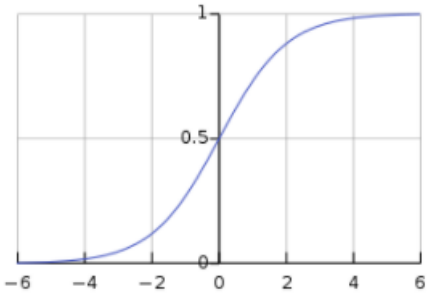
- 랜덤한 분포(일반적으로 가우시안 분포)를 입력으로 받고 이미지와 같은 데이터 출력
- 랜덤한 입력은 생성할 이미지의 잠재 표현(coding)으로 생각

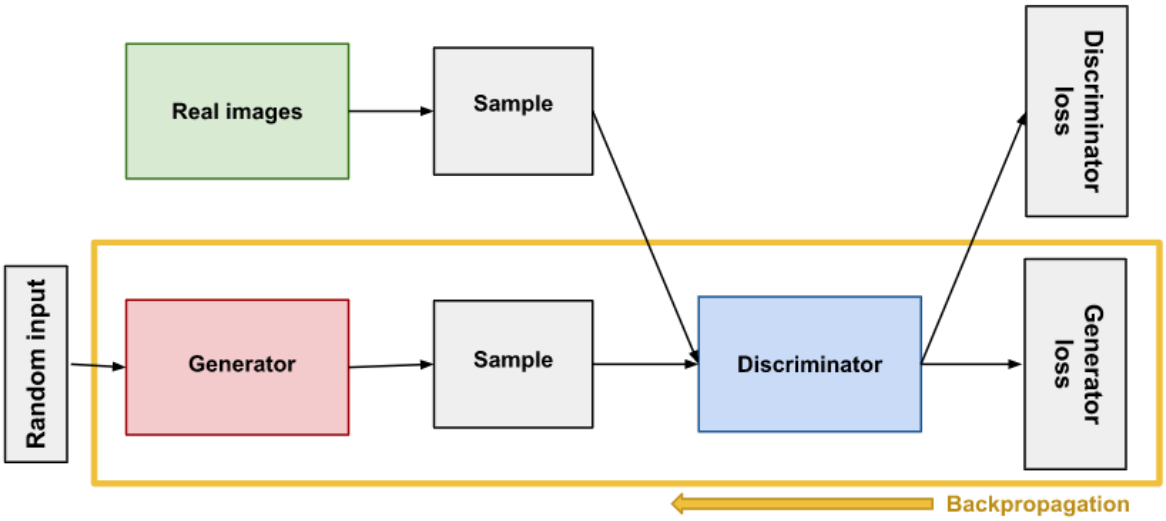


Discriminator

Goal : 생성자에서 얻은 가짜 이미지나 훈련 세트에서 추출한 진짜 이미지를 입력으로 받아 이미지가 가짜인지 진짜인지 구분

- 일반적인 이진분류



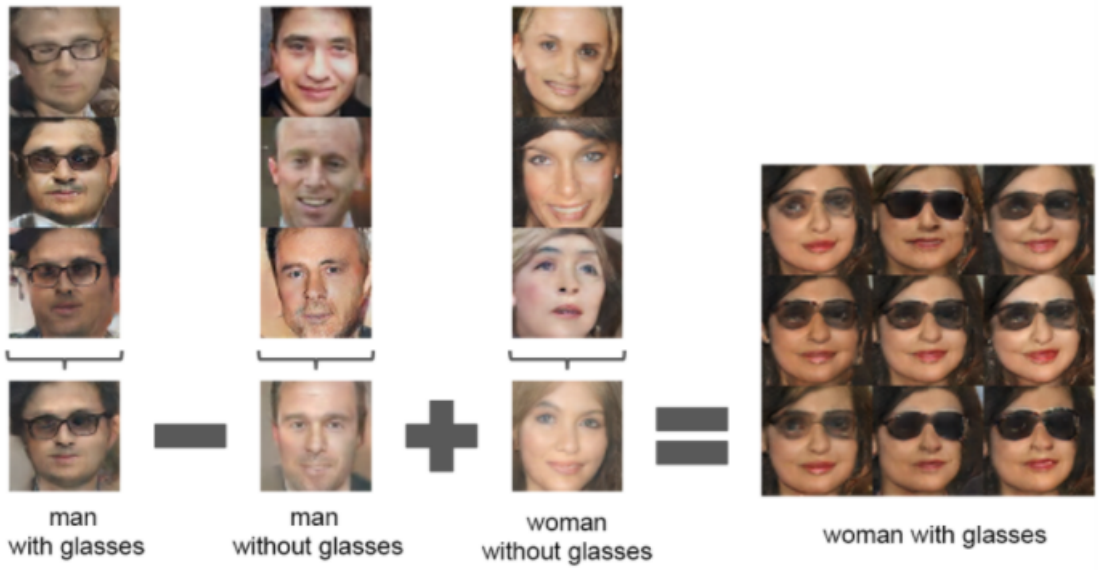


DCGAN, Deep convolution GAN

큰 이미지를 위해 deep conv layer을 기반으로 한 GAN 제작

안정적인 DCGAN?

> - 판별자에 있는 풀링 층을 스트라이드 합성곱으로 바꾸고 생성자에 있는 풀링 층은 전치 합성곱으로 바꿈
> - 생성자와 판별자에 배치 정규화 사용. 생성자의 출력층과 판별자의 입력층은 제외
> - 층을 깊게 쌓기 위해 완전 연결 은닉층 제거
> - `tanh` 함수를 사용해야 하는 출력층을 제외하고 생성자의 모든 층은 `ReLU` 활성화 함수 사용
> - 판별자의 모든 층은 `LeakyReLU` 활성화 함수를 사용

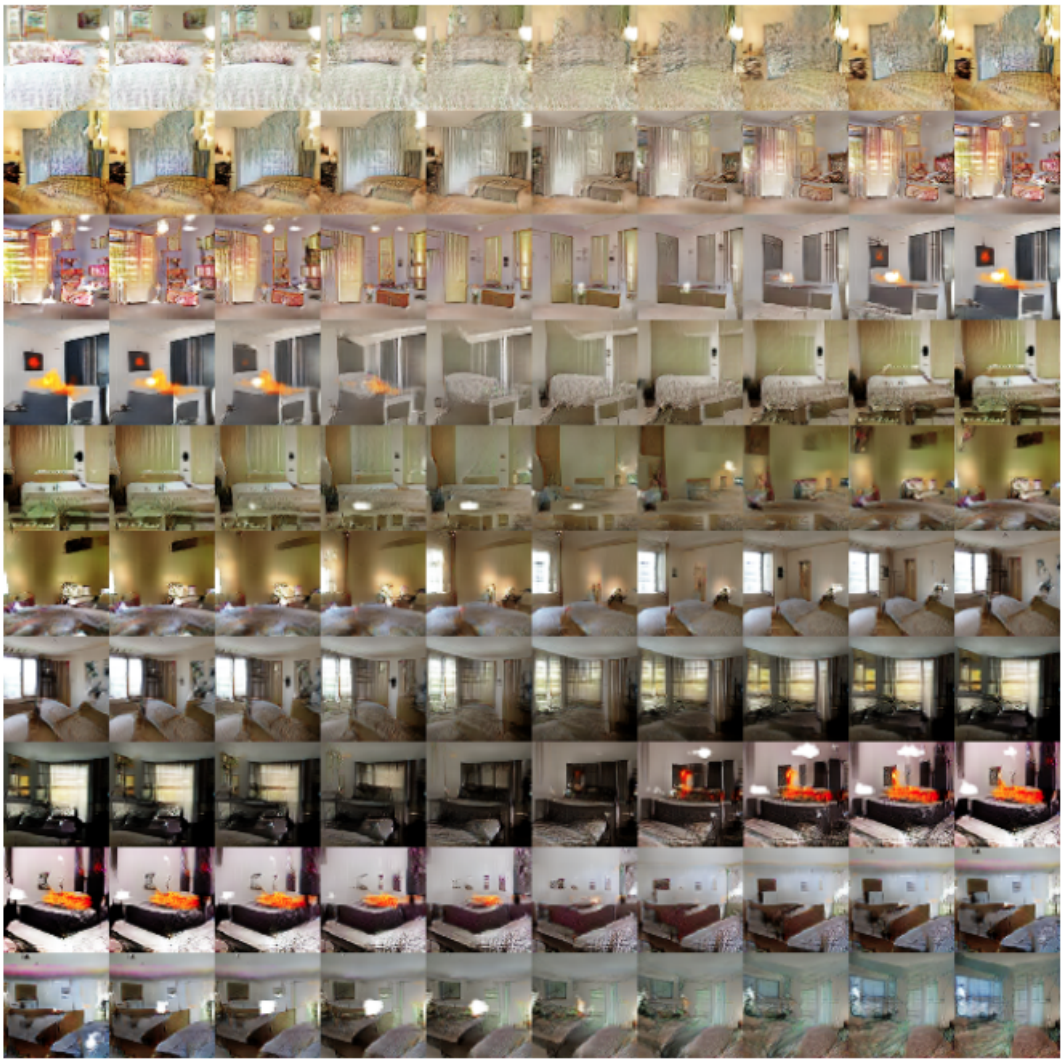


시각적 개념의 벡터 연산

기존의 word2vec 연구에서는 nn을 사용하여 말뭉치에서 실제로 단어 간의 관계를 학습하는 것을 보여주었고, DCGAN은 이를 말뭉치가 아닌 이미지에서 하는 것이 가능하다는 것을 시사

이미지가 갖는 의미를 바탕으로 직관적인 벡터 산술이 가능

- 안경을 쓴 남자와 안경을 쓰지 않은 남자, 안경을 쓰지 않은 여자를 생성하는 입력값들이 hidden layer에 벡터로 존재하는데, 각각의 벡터를 서로 빼고 더해주면 최종적으로 안경을 쓴 여자를 생성하는 입력 벡터를 찾는 것이 가능



hidden layer에서 돌아다니기

은닉 공간에서 벡터 연산이 가능하다는 것과 입력에 변화를 줬을 때 생성되는 결과가 부드럽게 변하는 것을 보는 분석이 중요한 이유는 GAN의 생성기가 단순한 mapping 함수를 학습한 것이 아니라는 것을 시사

DCGAN은 매우 큰 이미지를 생성하면 국부적으로 특징이 구분되지만 전반적으로 일관성 없는 이미지를 얻을 가능성 농후

ProGAN, PGGAN

- 훈련 초기에 작은 이미지를 생성하고 점진적으로 생성자와 판별자에 합성곱 층을 추가해 회차가 늘어날수록 큰 이미지를 만드는 방법

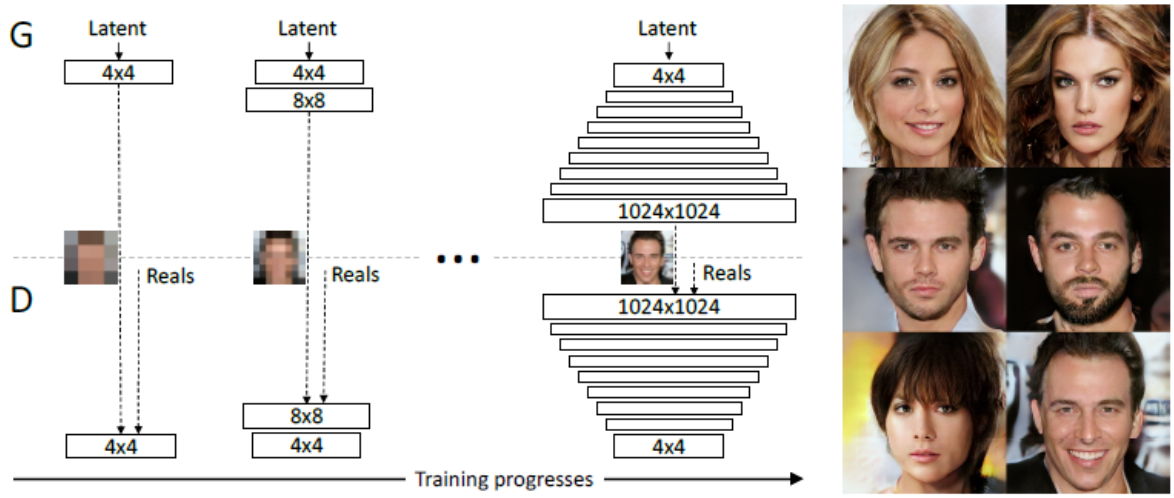


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

한 번에 전체 크기의 image feature들을 학습시키보다 4x4 저해상도를 시작함으로써 Large-scale structure를 찾아내도록 하고 점차 finer scale detail을 찾을 수 있도록 1024x1024 고해상도로 높아지는 것이 해상도적 측면에서 더 도움이 된다는 것

Mode Collapsing을 방지하는 방안

- 출력의 다양성 증가, 안정적인 훈련

미니배치 표준편차 층

생성자가 만든 이미지에 다양성이 부족하면 판별자의 특성 맵 간의 표준편차가 작을 것. 이 층 덕에 판별자는 통계를 쉽게 얻을 수 있고 다양성이 아주 적은 이미지를 만드는 생성자에게 속을 가능성 감소

--> 생성자가 조금 더 다양한 출력을 만들도록 유도하고 mode collapsing의 위험 줄임

동일한 학습 속도

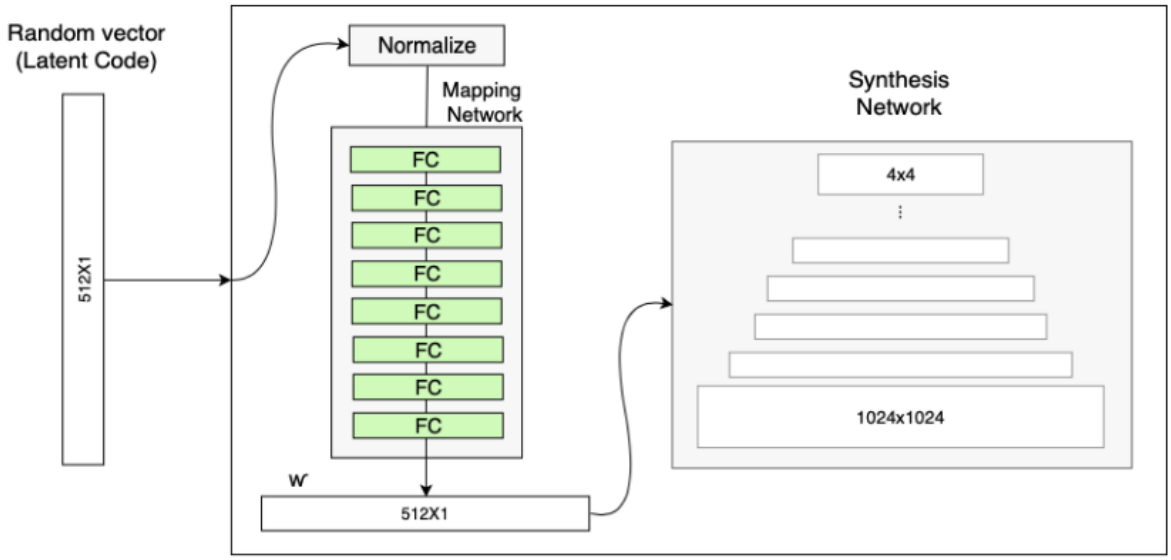
픽셀별 정규화 층

StyleGAN

Mapping Network

Goal : input vector의 각기 다른 시각적 특징을 다른 요소로 control 할 수 있는 중간 벡터로 인코딩

issue : 기존 GAN의 생성자는 그 특징들이 서로 얽혀있어서 벡터를 조절하면 얽힌 여러 특징이 동시에 변한다는 것 --> entanglement

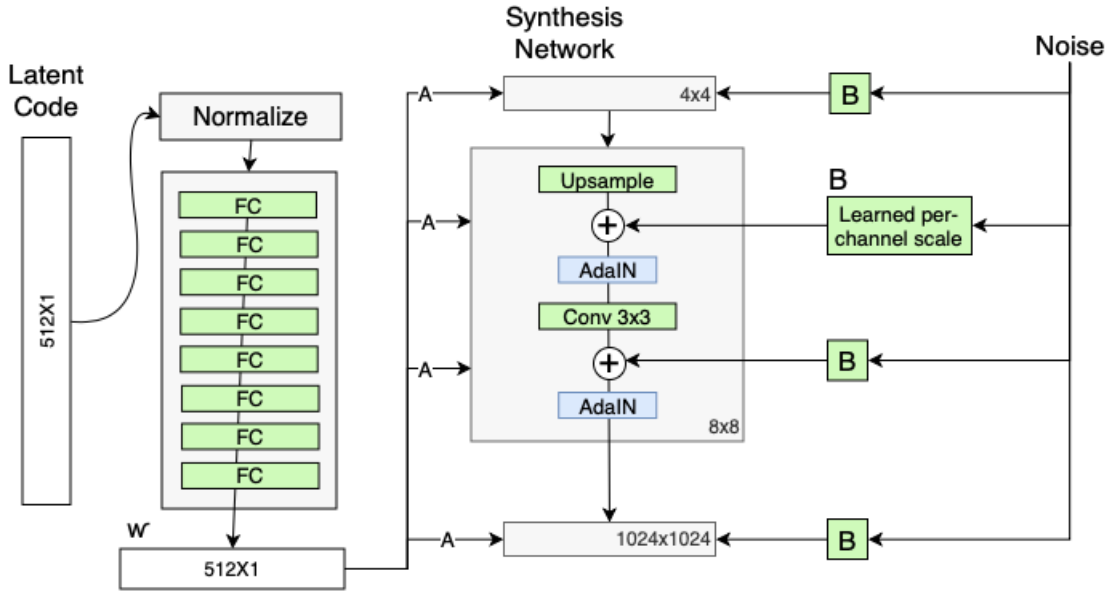


The generator with the Mapping Network (in addition to the ProGAN synthesis network)

Synthesis Network

이미지 생성을 담당. 일정하게 학습된 입력을 받아 이 입력을 합성곱 여러 개와 [upsampling] [https://m.blog.naver.com/PostView.nhn?blogId=worb1605&logNo=221266339261&proxyReferer=https:%2F%2Fwww.google.com%2F] 층에 통과시킴

- 입력과 (활성화 함수 전에 있는) 모든 합성곱 층의 출력에 잡음이 조금 섞임
- 잡음이 섞인 다음에 적응적 인스턴스 정규화 (AdaIN) 층이 뒤따름
- 각 특성 맵을 독립적으로 표준화한 다음 스타일 벡터를 사용해 각 특성 맵의 스케일과 이동을 결정



Adding scaled noise to each resolution level of the synthesis network

Mixing regularization(Style mixing)

StyleGAN 생성자는 합성 네트워크의 각 단계에서 중간 벡터를 이용하는데, 이로 인해 네트워크는 각 단계가 상관관계가 있음을 학습. 이러한 상관관계를 줄이기 위해 모델은 랜덤하게 두 개의 인풋 벡터를 선택한 후 각각에 대한 중간 벡터 w를 생성

coding c1과 c2가 mapping network를 지나며 두 style vector w1과 w2를 만듦. 이후 synthesis network의 첫 번째 단계에서 스타일 w1으로, 나머지 단계에서 스타일 w2를 바탕으로 이미지 생성

이는 **네트워크가 인접한 수준의 스타일이 상관관계를 가진다고 가정하지 못하도록 막음**

--> 각 스타일 벡터가 생성된 이미지에 있는 제한된 개수의 속성에만 영향을 미치는 styleGAN의 국지성 촉진

References

https://excelsior-cjh.tistory.com/187

https://ratsgo.github.io/generative%20model/2018/01/27/VAE/

https://aigong.tistory.com/65