DCGAN :
UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Deep Convolutional GAN을 사용한 비지도 표현 학습

2021210088 허지혜

Abstract

• 최근 몇 년 동안 CNN을 사용한 **지도 학습**은 Computer Vision 응용에서 엄청나게 많이 쓰였음. 상대적으로, CNN을 사용한 **비지도 학습**은 주목받지 못함.

• 위 논문을 통해 비지도학습과 지도학습을 위한 CNN의 격차를 해소되길 바람.

• 위 논문에서는 **Deep Convolutional GAN(DCGAN)**을 소개함.

• 다양한 이미지 데이터셋에서 학습하며 deep convolutional adversarial pair가 Generator와 Discriminator 모두에서 계층적인 표현을 학습(representation learning)한다는 것을 보여줌.

• 일반적인 이미지 표현문제와 같은 응용을 증명하기 위해 학습된 features를 사용함.

1. Introduction

- Convolutional GAN Architecture 형태에 대한 제약 조건들을 제안하고 평가하여 대부분의 setting에서 학습이 안정화되도록 함.
- 이러한 구조의 분류를 Deep Convolutional GAN이라고 함.
- 이미지 분류 문제를 위해 학습된 Discriminator를 사용하고 다른 비지도 학습 알고리즘들에 경쟁적인 성능을 보여줌.
- GAN에 의해 학습된 filters를 시각화하고 특정한 filters이 특정한 객체들을 학습하는 것을 보여줌.
- Generator가 생성한 샘플 데이터가 흥미로운 vector 산술 특성을 가지고 있음을 보여줌.

2. Related work

2.1) Representation Learning From Unlabeled Data

- 비지도 표현 학습의 전통적인 접근법

-> 데이터를 클러스터링, 분류 스코어 향상에 클러스터의 이점 활용

- 이를 이미지 맥락으로 보면,

이미지 표현을 학습하기 위해 이미지 패치의 계층적 클러스터링 하는 것이 가능함.

2. Related work

2.2) Generating Natural Images

- Generative image models은 두 가지 범주로 나뉩니다.

nonparametric

- nonparametric 모형은 종종 존재하는 이미지의 데이터베이스로부터로 되고(?) 이미지 패치, 텍스처 합성, 초해상도, painting 분야에서 사용되어져 옴.

parametric

- Parametric 모형은 광범위하게 탐색됨.
- Variational sampling, iterative forward diffusion process등이 있었지만 Natural한 이미지를 생성하는 것에 대하여 실제와 비슷하게 성공하지는 못했음.
- Recurrent network, Deconvolutional network는 Natural한 이미지를 생성하는데 약간의 성공을 이루었지만 지도학습 문제를 위해 Generator를 활용하지는 않음.

2. Related work

2.3) Visualizing the internals of CNNs

- NN에 대한 가장 큰 비판은 network가 어떻게 작동하는지 모르는 **Black-box** 방식이라는 것임.

-> context of CNN(Zeiler et. al. (Zeiler & Fergus, 2014))에서는 Deconvolution과 maximal activation을 filtering 하는것을 보여주었고, 목적을 대략 찾을 수 있음.

-> 비슷하게, 입력에 GD(Gradient Descent)를 사용하는 것은 특정한 filters의 부분집합을 활성화하여 이상적인 이미지를 검사할 수 있음. (Mordvintsev et al.)

3. Approach and Model Architecture

- 앞서, GAN에는 많은 문제들이 있었는데 그 중 하나는 고해상도 이미지를 얻지 못함.
- 대부분 노이즈가 끼거나 흐렸음.

-> GAN에 CNN을 사용하여 scale-up하는 시도를 함.
-> LANPGAN이 대표적이었는데 LANPGAN 논문의 저자로부터 동기를 얻음.

넓은 범위의 dataset에 대해 안정적인 결과를 얻었고 고해상도의 이미지와 깊은 Generator 모형을 학습할 수 있도록 하는 Architecture를 찾음.

DCGAN

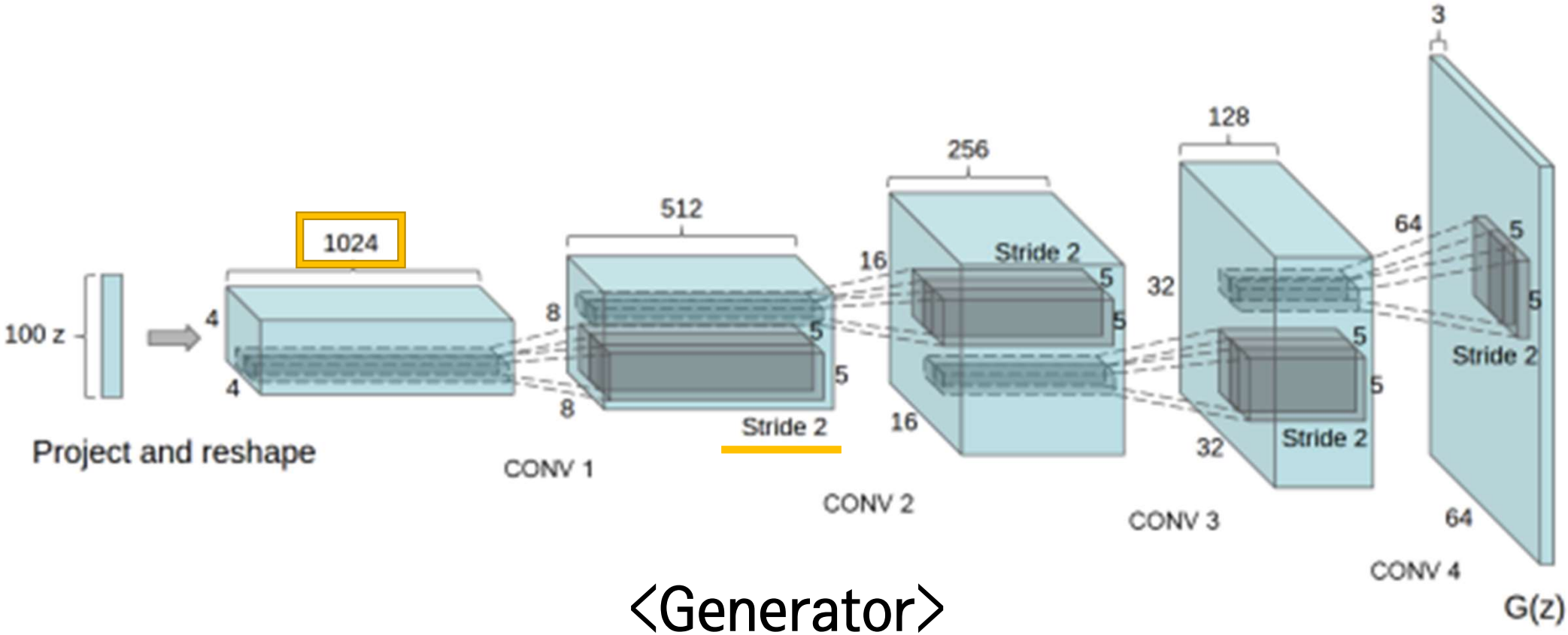
3. Approach and Model Architecture

- DCGAN은 GAN과 구조가 비슷하지만 큰 몇 가지 차이점이 있음.

- Pooling layer -> large stride, convolutional layer
  -> Stride를 이용하여 이미지 크기를 줄임. Generator에서는 일반적인 layer가 아닌 franctional strided Convolutional layer로 바꾸는데 이는 input에서 padding을 추가하여 가로 세로 크기를 늘리는 것임.
- Generator/ Discriminator Batchnorm 적용
  -> Generator의 output, Discriminator의 input 부분 제외 BatchNorm 적용

- Fully-Connected layer 제거
  -> 너무 많은 파라미터로 계산량이 증가하기 때문임.
  근데 Generator의 첫 부분과 Discriminator의 마지막 부분에 FC layer가 하나씩 있음.
- Generator activation function : ReLU, output use tanh
- Discriminator activation function : LeakyReLU

4. Details of Adversarial Training



<Generator>

4. Details of Adversarial Training

- DCGAN은 LSUN, ImageNet, Face dataset에 대하여 학습을 함.
  - TRAIN DATA에 대하여 전처리는 따로 안함.
  - tanh activation 함수는 [-1,1] scaling
  - 모든 모형은 mini-batch 128으로 학습함.
  - GAN에서는 momentem을 이용, DCGAN에서는 Adam 이용
  - Learning rate = 0.002 제안
  - 가중치 초기화 ~N(0,0.02)
  - LeakyReLU alpha=0.2

4. Details of Adversarial Training

4.1) LSUN

- Generator 모형으로부터 학습 샘플에 대한 overfitting과 memory 문제를 걱정.
- 하지만 이 dataset을 통해 위 문제를 통해 이미지를 생성해내는 것이 아님을 설명함.
- 3백만장 이상의 학습 샘플이 포함되어 있는 LSUN dataset에 모형을 학습함.
- 이때, 이미지 **중복 제거 과정**을 수행함. (denoising autoencoder 사용)
- -> 275,000 장 정도의 중복 이미지를 제거

4. Details of Adversarial Training

4.1) LSUN

한 에폭으로 학습시키고 생성한 이미지

Memory 문제가 일어나지 않았다는 증거.
학습률도 작게 설정하고 mini-batch SGD로 했기 때문에 학습 이미지를 기억할 수 없음.

4. Details of Adversarial Training

4.1) LSUN

다섯 에폭으로 학습시키고 생성한 이미지

Overfitting 문제보다 오히려 Underfitting이 일어났다고 말함.

침대의 머리 부분에 약간의 노이즈가 반복된다고 주장.

4. Details of Adversarial Training

4.2) Faces

- 웹에서 현대 출생 사람 위주로 모음. 1만명의 사람에 대해 3백만장 이미지.
- OpenCV Face detector를 이용하여 350,000개의 Face box를 제공하면서 고해상도 이미지를 검출한 데이터를 학습에 사용함.
- 따로 data augmentation은 사용하지 않음.

4.3) ImageNet - 1K

- ImageNet-1K를 비지도 학습을 위한 Natural 이미지로 사용함.
- 32x32 min-resized centor crops로 학습함.
- 따로 data augmentation은 사용하지 않음.

5. Empirical Validation of DCGANs Capabilities

5.1) Classifying CIFAR-10 using GANs as a FEATURE EXTRACTOR

- 비지도 표현 학습 알고리즘을 평가하는 일반적인 방법 중 하나는 Discriminator는 입력 이미지의 특징을 파악할 수 있도록 filter가 학습되는데, 이 filter를 이용해 지도학습을 진행한 결과를 보는 것임.
- ImageNet-1K(train), 학습된 Discriminator feature를 CIFAR-10 dataset을 이용하여 test한 결과를 보여줌.
- CIFAR-10 dataset을 이용하여 성능을 평가하면 다음과 같은 표가 나옴.

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% (±0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant K-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% (±0.4%)	512

5. Empirical Validation of DCGANs Capabilities

5.2) Classifying SVHN digits using GANs as a FEATURE EXTRACTOR

- SVHN(StreetView House Numbers) dataset(Netzer et al, 2011)
- DCGAN을 ImageNet-1k dataset에 pretrained 한 뒤, 위 1)과 동일한 방식으로 분류 task를 수행함.

-> 당시 엄청난 성능을 달성

-> 같은 architecture에 purely Supervised CNN을 사용했을 때 28.87%를 보임.

-> 따라서 성능이 '모델 구조' 때문은 아님을 발견.

Table 2: SVHN classification with 1000 labels

Model	error rate
KNN	77.93%
TSVM	66.55%
M1+KNN	65.63%
M1+TSVM	54.33%
M1+M2	36.02%
SWWAE without dropout	27.83%
SWWAE with dropout	23.56%
DCGAN (ours) + L2-SVM	22.48%
Supervised CNN with the same architecture	28.87% (validation)

6. Inverstigating and Visualizing the Internals of the Networks

6.1) Walking in the Latent Space

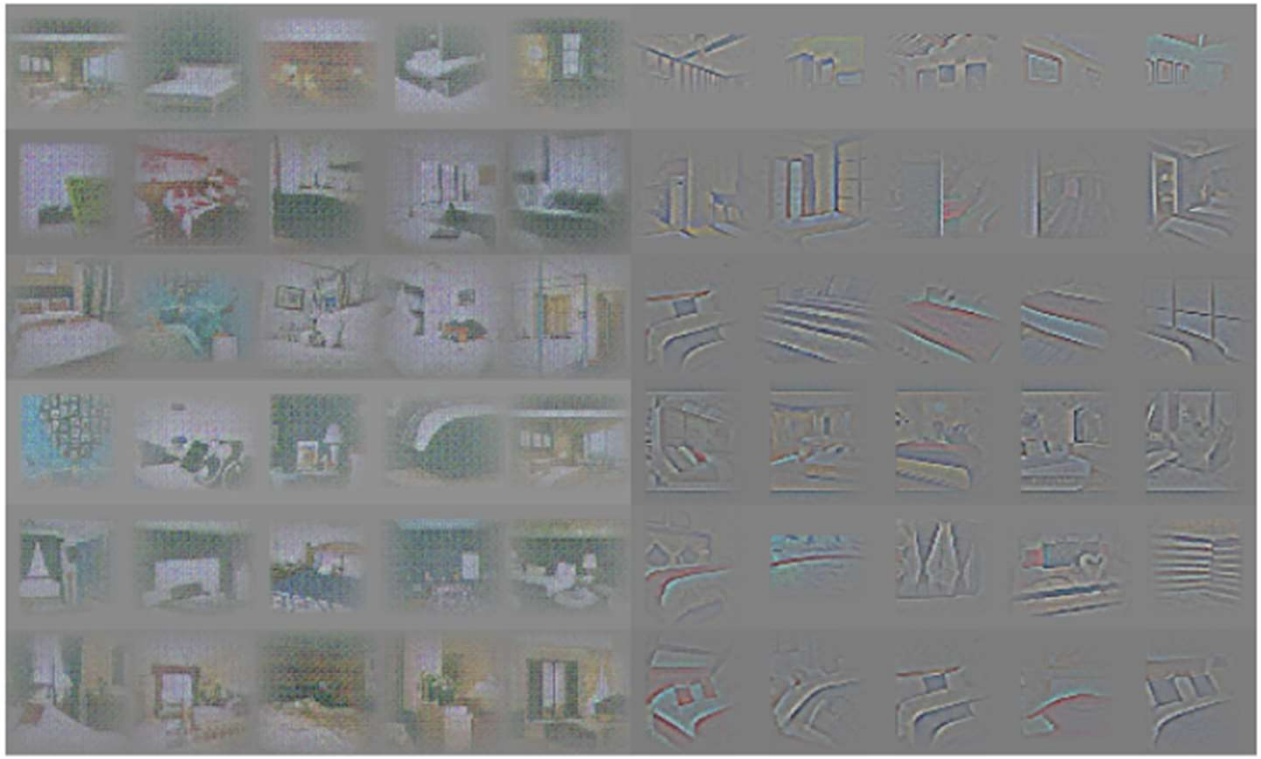
- 학습된 Latent space(잠재 공간)

: 이를 살펴보는 것은 모델이 학습한 표현에 대해 다양한 통찰을 줄 수 있음.

- 생성되는 이미지를 살펴보면 다음과 같이 smooth하게 물건이 생기거나 가구의 방향/형태가 바뀌는 등의 현상을 관찰할 수 있음.

6. Inverstigating and Visualizing the Internals of the Networks

6.2) Visualizing the Discriminator Features

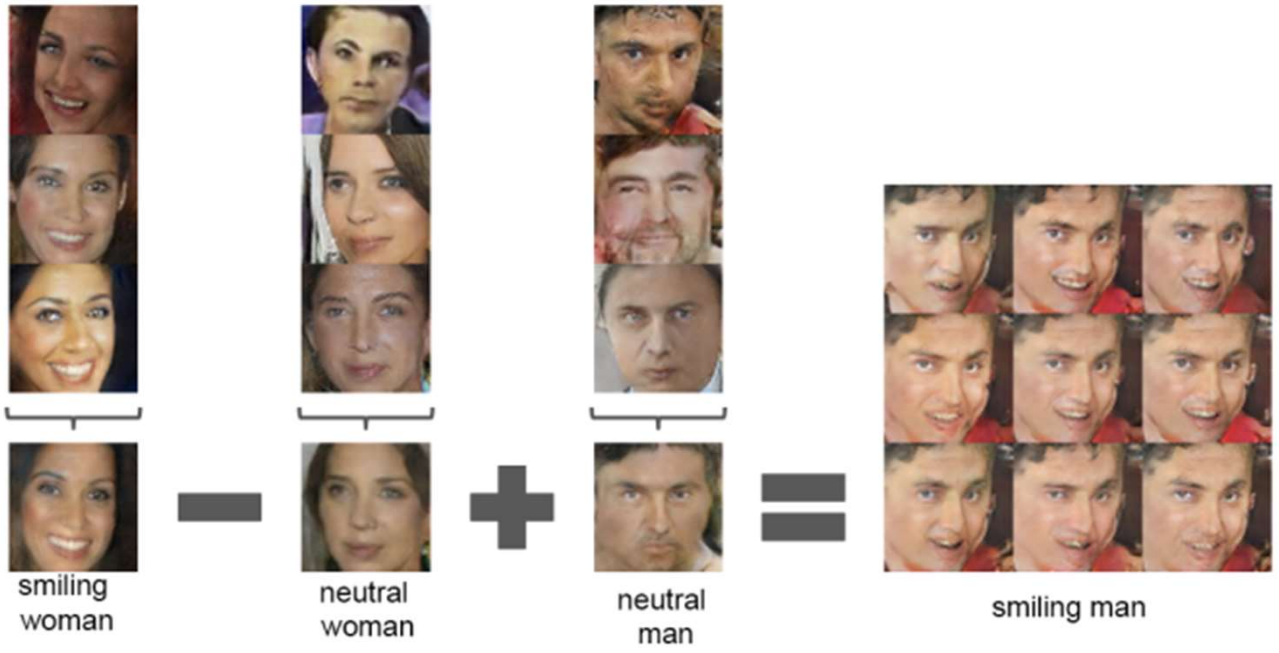


Random filters Trained filters

- 비지도학습으로 학습된 DCGAN 역시 강력한 계층적인 Features를 학습할 수 있다는 것을 보여줌.
- Guided Backpropagation을 사용해 discriminator의 학습된 filters가 activative되는 부분을 시각화한 모습.

-> 이를 통해 각 filter가 나름 이미지의 특징을 잘 이해했다고 봄.

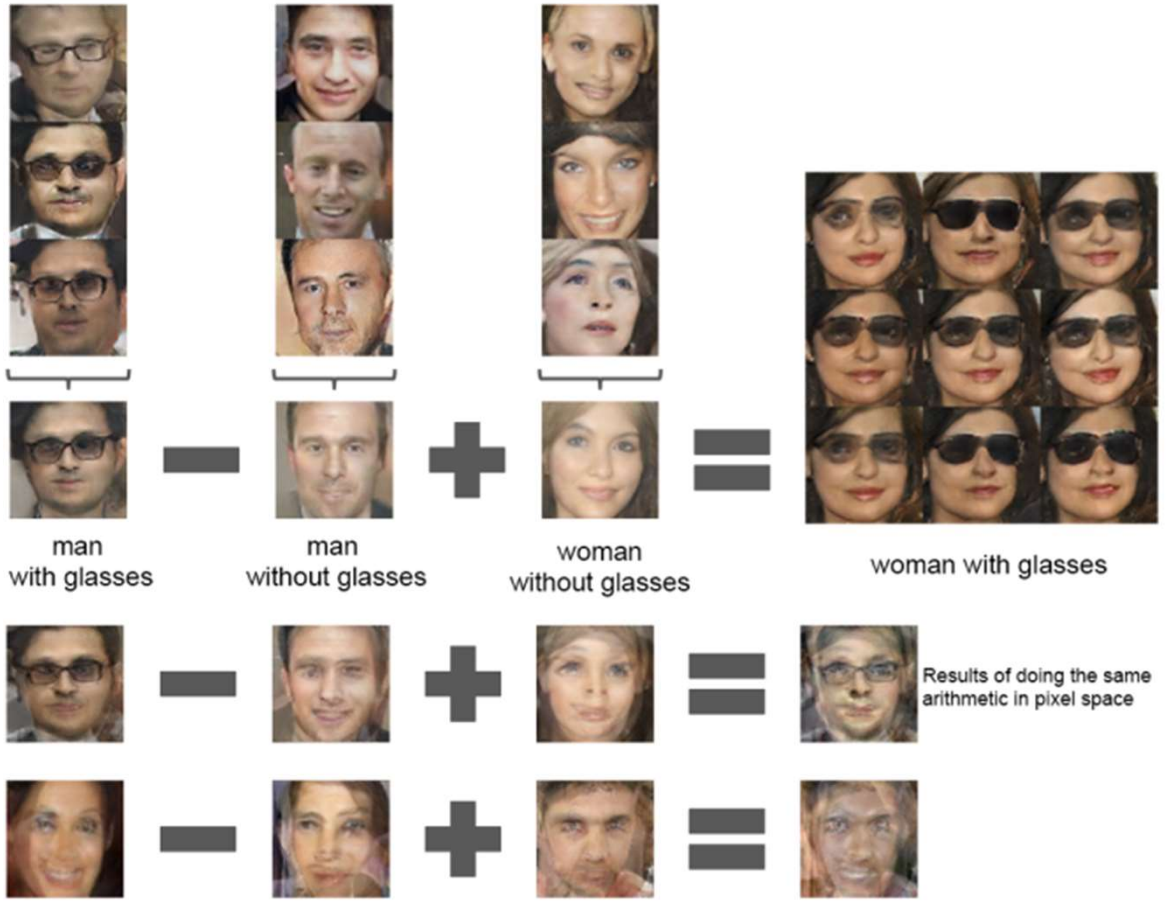
7. Conclusion and Future Work



- Generator가 만든 이미지임.
- Network가 무언가를 **이해**하고 있다는 것을 알 수 있음.
- (Smiling, Woman) + (Neutral, Woman) - (Neutral, man) = **(Smiling, man)**

- 자연어 처리에서는 word2vec이고 이미지 분야에서는 DCGAN을 이용하여 할 수 있게 됨.

7. Conclusion and Future Work



7. Conclusion and Future Work

- **이해**하고 있다는 또 다른 증거.
- 왼쪽 얼굴 - z_left vector
- 오른쪽 얼굴 - z_right vector
- 두 벡터의 사이를 잇는 축을 interpolatin하여 generator에 넣어보니 천천히 turn하는 얼굴이 나오는 것을 볼 수 있음.

7. Conclusion and Future Work

- DCGAN은 GAN을 학습하기 위해 더 안정적인 아키텍처를 제안하고 Discriminator Network가 지도학습과 Generative modeling을 위한 이미지 표현도 잘 학습할 수 있다는 증거를 제공함.
- 하지만 여전히 모형의 불안정성이 남아있음. 이를 해결할 연구가 필요하다고 봄.
- 비디오나 오디오 등 다른 도메인에 확장시켜보는 연구도 좋을 것 같음.

Code

Generator

```
def generator(x, reuse=False):
    with tf.variable_scope('Generator', reuse=reuse):
        # TensorFlow Layers automatically create variables and calculate their
        # shape, based on the input.
        x = tf.layers.dense(x, units=7 * 7 * 128)
        x = tf.layers.batch_normalization(x, training=is_training)
        x = tf.nn.relu(x)
        # Reshape to a 4-D array of images: (batch, height, width, channels)
        # New shape: (batch, 7, 7, 128)
        x = tf.reshape(x, shape=[-1, 7, 7, 128])
        # Deconvolution, image shape: (batch, 14, 14, 64)
        x = tf.layers.conv2d_transpose(x, 64, 5, strides=2, padding='same')
        x = tf.layers.batch_normalization(x, training=is_training)
        x = tf.nn.relu(x)
        # Deconvolution, image shape: (batch, 28, 28, 1)
        x = tf.layers.conv2d_transpose(x, 1, 5, strides=2, padding='same')
        # Apply tanh for better stability - clip values to [-1, 1].
        x = tf.nn.tanh(x)
        return x
```

- Input : Noise
- Output : Image

Code

Discriminator

```
def discriminator(x, reuse=False):
    with tf.variable_scope('Discriminator', reuse=reuse):
        # Typical convolutional neural network to classify images.
        x = tf.layers.conv2d(x, 64, 5, strides=2, padding='same')
        x = tf.layers.batch_normalization(x, training=is_training)
        x = leakyrelu(x)
        x = tf.layers.conv2d(x, 128, 5, strides=2, padding='same')
        x = tf.layers.batch_normalization(x, training=is_training)
        x = leakyrelu(x)
        # Flatten
        x = tf.reshape(x, shape=[-1, 7*7*128])
        x = tf.layers.dense(x, 1024)
        x = tf.layers.batch_normalization(x, training=is_training)
        x = leakyrelu(x)
        # Output 2 classes: Real and Fake images
        x = tf.layers.dense(x, 2)
    return x
```

- Input : Image
- Output : Real/Fake image Prediction

Reference

[1] DCGAN 논문 : https://arxiv.org/pdf/1511.06434.pdf

[2] DCGAN 코드 : https://github.com/eriklindernoren/Keras-GAN/blob/master/dcgan/dcgan.py

[3] DCGAN 논문 리뷰 블로그 :

https://mole-starseeker.tistory.com/21

https://velog.io/@changdaeoh/DCGAN

https://ysbsb.github.io/gan/2020/12/05/DCGAN.html

https://roytravel.tistory.com/109

https://kau-deeperent.tistory.com/79

끝