GAN / DCGAN 리뷰

Jung Choi

Generative model

"What I cannot create, I do not understand."

- **Generative Methods**
  - Model **class-conditional pdfs** and **prior probabilities**
  - "Generative" since sampling can generate synthetic data points
  - Popular models
    - Gaussians, Naïve Bayes, Mixtures of multinomials
    - Mixtures of Gaussians, Mixtures of experts, Hidden Markov Models (HMM)
    - Sigmoidal belief networks, Bayesian networks, Markov random fields

- **Discriminative Methods**
  - **Directly estimate posterior** probabilities
  - No attempt to model underlying probability distributions
  - Focus computational resources on given task– better performance
  - Popular models
    - Logistic regression, SVMs
    - Traditional neural networks, Nearest neighbor
    - Conditional Random Fields (CRF)

http://www.cedar.buffalo.edu/~srihari/CSE574/Discriminative-Generative.pdf

Why generative model?

NIPS 2016 GAN tutorial 에 따르면..

- High dimension에서 probability distribution을 추출해서 다룰 수 있게 됨. : Latent feature dimension에 대한 이해가 생기기 때문에 그 레벨에서 재미있는 작업들이 가능해진다는 뜻.

- 강화학습에서 model-based model들은 generative model을 그 안에 포함한다. (sequential한 데이터에 대한 generative model이 미래의 데이터를 예측할 수 있게 해줌.)

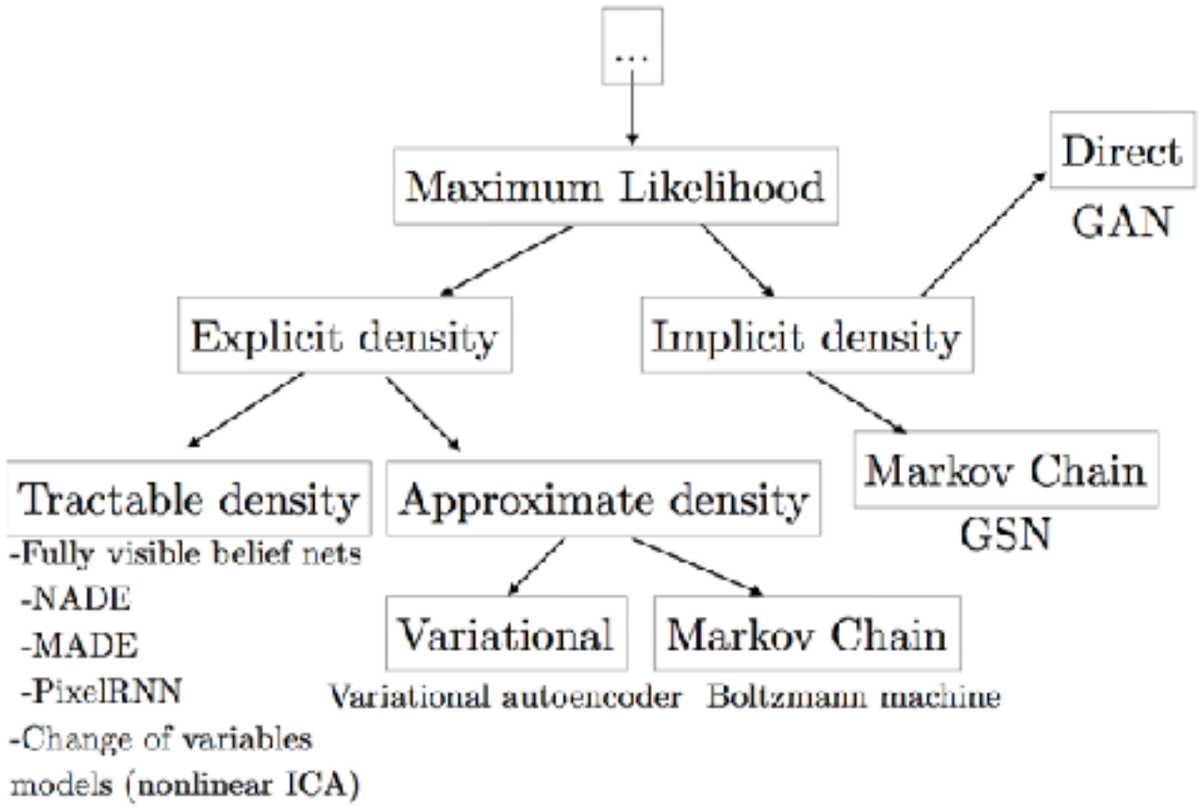
- Semi-supervised learning에서 활용될 수 있다. (supervised에 필요한 샘플을 generate해서 쓴다던지..)

NIPS 2016 Tutorial: GAN (https://arxiv.org/abs/1701.00160)

GAN은 generative model입니다.

쉽게 얘기하자면, 많은 양의 데이터를 통해 학습해서 그와 비슷한 데이터를 생성할 수 있는 모델을 만드는 것.

근데, 우리의 **뉴럴넷**은 이 데이터를 생성하는 **확률모델의 파라메터를 학습**하거나(DBN, VAE 등), 데이터 생성모델 **자체를 implicit하게 학습(내지는 구현이랄까?)**하기도 한다. GAN은 후자.



NIPS 2016 Tutorial: GAN (https://arxiv.org/abs/1701.00160)

GAN

- 인풋으로 uniform distribution으로부터 일정 갯수의 random number를 받아서 >> 아웃풋으로 특정 사이즈의 이미지를 생성하는 모델.

- 이 random input이 달라지면 생성되는 이미지도 달라짐. (결국 random input이 어떤 condition의 역할을 하는 latent feature의 역할이고, 이미지를 생성하기 위한 특성들을 담는 feature의 트리거가 되는 역할이기 때문에, 우리가 원했던 generative model이라고 여길 수 있다.)

- Adversarial하게 학습하는 디테일은 지난 시간에 재미있게(?) 봤습니다.

Why gan?

GAN은 다른 generative model들의 단점을 피하기 위해 고안되었다.

- **FVBN**이 x의 dimension에 비례한 runtime을 필요로 하는데 비해서,(이전 sample에 condition되어 다음 sample 생성) GAN은 sample들 병렬적으로(in parallel, 즉 한방에) 생성할 수 있다.

- **Boltzmann machine**이 generator function을 디자인함에 있어 제약사항을 두는 것에 비해서 GAN은 제한 사항이 거의 없다. (Boltzmann machine에서는 tractable한 Markov chain sampling을 위해 사용할 수 있는 prob. distribution의 종류가 제한되어 있다.(:? contrastive divergence 때를 얘기하는 건지?)

- 또한, **Nonlinear ICA**는 generator function이 invertible하고 latent z가 sample x와 같은 dimension을 가져야 하지만, GAN은 그렇지 않다.

  (잠깐, Nonlinear ICA ? 간단히 얘기하자면, 인풋 space를 generator space로 쏙 보내버림으로써 generator의 distribution을 구하겠다는 창의적인 발상. : latent variable z vector와 continuous, 미분 가능 & invertible한 transformation g가 있다고 할때, g(z)는 x space로부터 샘플을 뽑아낼 수 있다.)

NIPS 2016 Tutorial: GAN (https://arxiv.org/abs/1701.00160)

Why gan?

GAN은 다른 generative model들의 단점을 피하기 위해 고안되었다.

- Markov chain이 필요치 않다. 이것은 Boltzmann machine과 Generative stochastic network에 비해 장점이다.

- Variational bound를 필요로 하지 않고, GAN framework에서 사용할수 있는 model의 종류들은 이미 universal approximator로 알려진 모델들이기 때문에(무슨 의미?) GAN은 점근적으로 consistent하게 된다. 몇몇 VAE들도 점근적으로 consistent하게 될 것이라고 추측되긴 하지만 증명되지는 않았다.

- GAN은 다른 방법들에 비해 더 나은(주관적으로) 결과물을 내어준다고 여겨진다.

하지만 GAN에게도 단점이 있다. Training할 때 (D와 G 사이의 game에서) Nash 균형이 필요하다. 이것은 objective function을 최적화하는 문제보다 더 어려운 문제이다.

NIPS 2016 Tutorial: GAN (https://arxiv.org/abs/1701.00160)

Again, gan(dcgan) is a **generative model**

- 어떤 input distribution을 우리가 원하는 데이터의 (예를 들어 이미지 데이터의) distribution의 space로 맵핑하는 것.

- 근데 이 작업을 neural network로(DCGAN은 CNN으로) 해결하겠다는 것.

- 그니까, 학습 전의 랜덤한 weight값들로 된 DCGAN은 랜덤 인풋을 받아서 이미지 같지 않은 완전 랜덤한 이미지를 만들어 낼 것이고,

- 학습 후의 DCGAN은 랜덤 인풋을 받아서 이미지 space에서의 랜덤한 이미지, 즉 그럴 듯한 이미지를 만들어 낼 것.

DCGAN(Deep convolutional GAN)

- 왜 2015년말에 DCGAN, DCGAN 했을까? 대강 두가지 이유같다..

  1) 일단 보기에 신기해서

  2) break-through

- Break-through라 함은, 원래 GAN에 CNN을 도입하는 시도들이 성공적이지 못했는데, 3가지(+1) 변화를 통해서 도입에 성공했다는 것. 어떤 변화들?

  (1) All convolutional net.

  (2) Convolutional feature 위에 fully connected layer를 제거하는 방식.

  (3) Batch normalization.

  (+1) ReLU activation.

(1) All convolutional net

- Deterministic한 spatial pooling function(max-pooling같은)를 **strided convolution**으로 대체 -> 네트워크가 알아서 spatial한 down sampling을 학습하도록 함.
- 이를 generator에도 적용해서 스스로 spatial한 upsampling을 학습하도록 함.

Striving for Simplicity: The All Convolutional Net (https://arxiv.org/abs/1412.6806)

(2) Convolutional feature 위에 fully connected layer 를 없애봤어요.

- 이에 대한 대표적인 예는 image classification의 state-of-the-art CNN 모델들에서 사용된 global average pooling 방식이다.

- GAP이란?: 각 feature map의 average값을 취한 후, 이것들의 벡터에다가 바로 softmax를 적용하는 것.

- 잠깐, classification 문제에 있어서 GAP(질?)의 advantage :

  1) FCN을 거치지 않으므로 convolution structure에 연관성 깊게 학습. >> feature map과 class 사이의 관계가 더 direct 해짐.

  2) optimize할 파라메터가 없기 때문에, 적어도 이 레이어에서는 overfitting의 위험이 감소한다.

  3) spatial한 정보를 sum up 하니까 이미지들의 spatial한 translation에 더 robust해진다.

Network In Network, Min Lin et al. (https://arxiv.org/pdf/1312.4400.pdf)

(2) Convolutional feature 위에 fully connected layer 를 없애봤어요.

- 우리는 이 global average pooling이 **stability는 증가**시켜줬지만, **convergence 속도는 감소**시켰다는 것을 발견했다.

- 따라서 마지막 convolutional feature를 직접 generator와 discriminator의 input과 output에 각각 연결하는 방식 (나름 중도적인 방식)을 썼는데, 좋은 결과를 가져다 주었다.

- GAN의 첫 레이어(uniform distribution에서의 noise z를 인풋으로 받는)는 fully connected layer라고 여겨질 수 있다.(just 행렬곱이기 때문에) 하지만 결과값은 4-dimensional tensor로 reshape되고, convolution task의 출발지점으로 사용된다.

- Discriminator에서는, 마지막 convolution 레이어가 flatten되고 하나의 sigmoid output에 넣어진다.

(3) Batch normalization

- 네트워크의 각 유닛에 들어가는 input을 normalize해서 0 mean / unit variance를 갖게 하는 방식. -> 학습을 stabilize하는 효과

- Initialization이 잘 되지 않은 경우에도 학습이 잘 이루어지도록 돕는다.

- Deep한 모델에서도 gradient flow가 잘 이루어지도록 돕는다.

- 이 방법이 deep generator가 제대로 작동하도록 하는데 핵심 역할을 했다. 보통 GAN모델에서 많이 일어나는 문제 – generate된 샘플들이 한 point에만 집중되어 나타나는 문제 – 를 방지해주는 역할을 했다.

- 모든 레이어에 batch normalization을 바로 적용하는 것은 ‘sample oscillation’문제와 불안정성을 야기했다. 하지만 generator의 output과 discriminator의 input에만 batch normalization을 적용하면 이런 문제가 일어나지 않았다.

(+1) ReLU activation

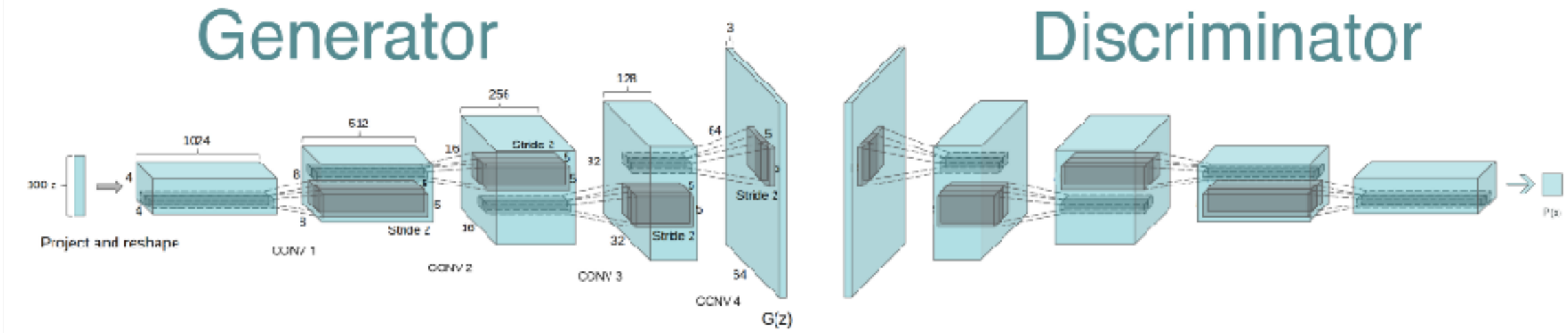
- 그 외, 최종 아웃풋 레이어 (Tanh function 사용)를 제외하고는 ReLU activation을 사용했다.

- 마지막 레이어에서 Tanh를 사용한 이유 : Bounded activation을 사용하는 것은 training distribution의 color space를 모두 커버하고 saturate하도록 더 빨리 학습하게 한다는 것을 발견했다.

- Discriminator에서 leaky ReLU를 사용했는데, 특히 resolution이 높은 모델에 더 효과적이었다.

- 참고로 original GAN에서는 max-out activation을 사용했다.

DCGAN 네트워크 구조



<Architecture guidelines for stable Deep Convolutional GANs>

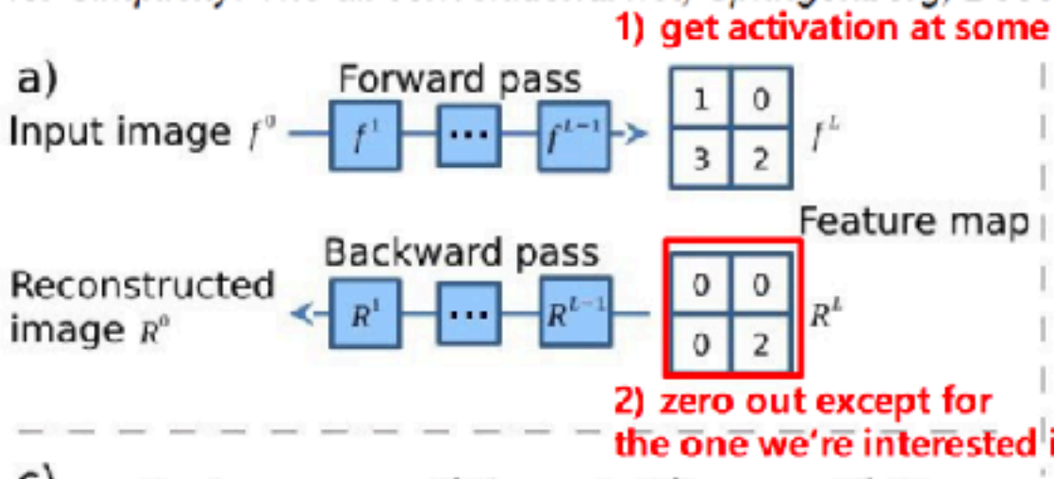
1. Replace any pooling layers with strided convolutions(discriminator) and **fractional-strided convolutions** (generator).
2. Use batchnorm in both the generator and the discriminator.
3. Remove fully connected hidden layers for deeper architectures.
4. Use ReLU activation in generator for all layers except for the output, which uses Tanh.
5. Use LeakyReLU activation in the discriminator for all layers.

잠깐, fractional-strided convolution?

Remember in cs231n?..

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

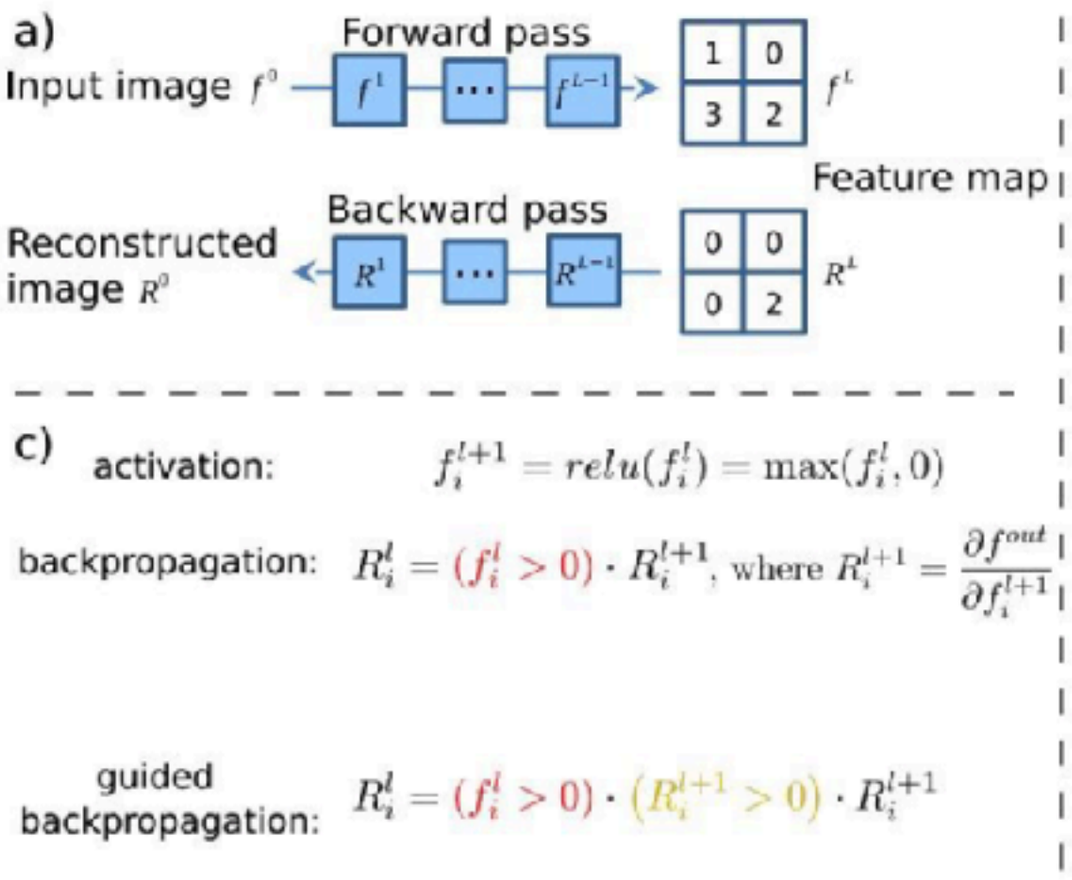


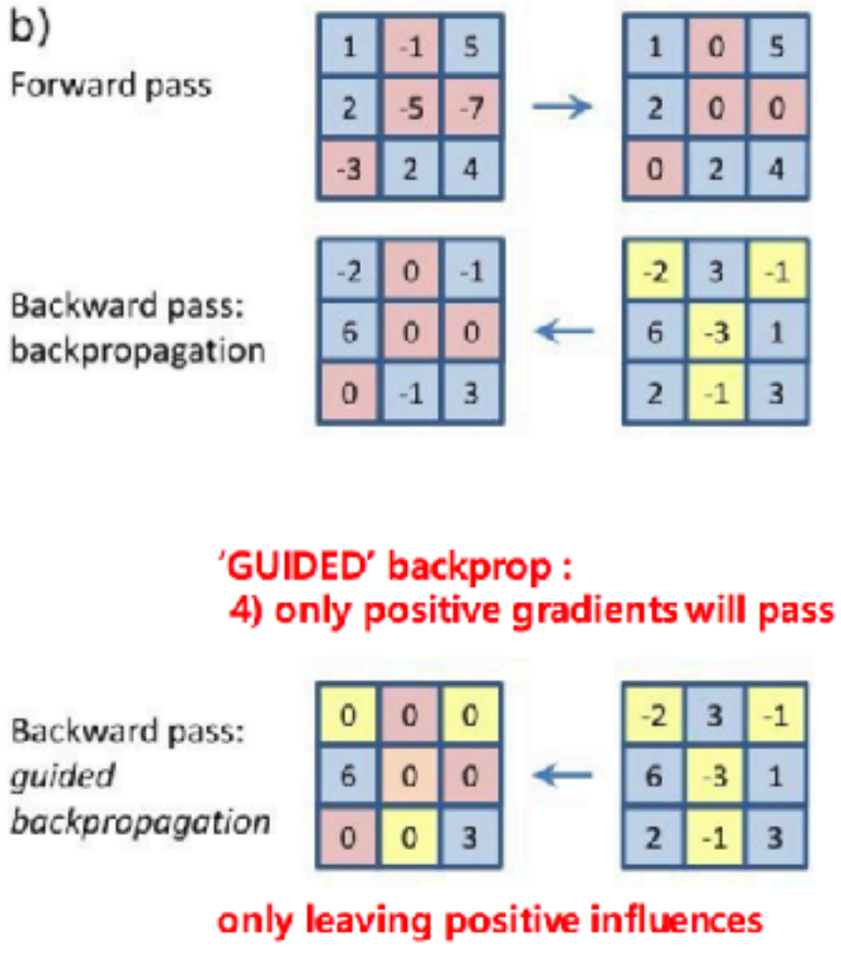
Backward pass for a ReLU (will be changed in Guided Backprop)

잠깐, fractional-strided convolution?

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



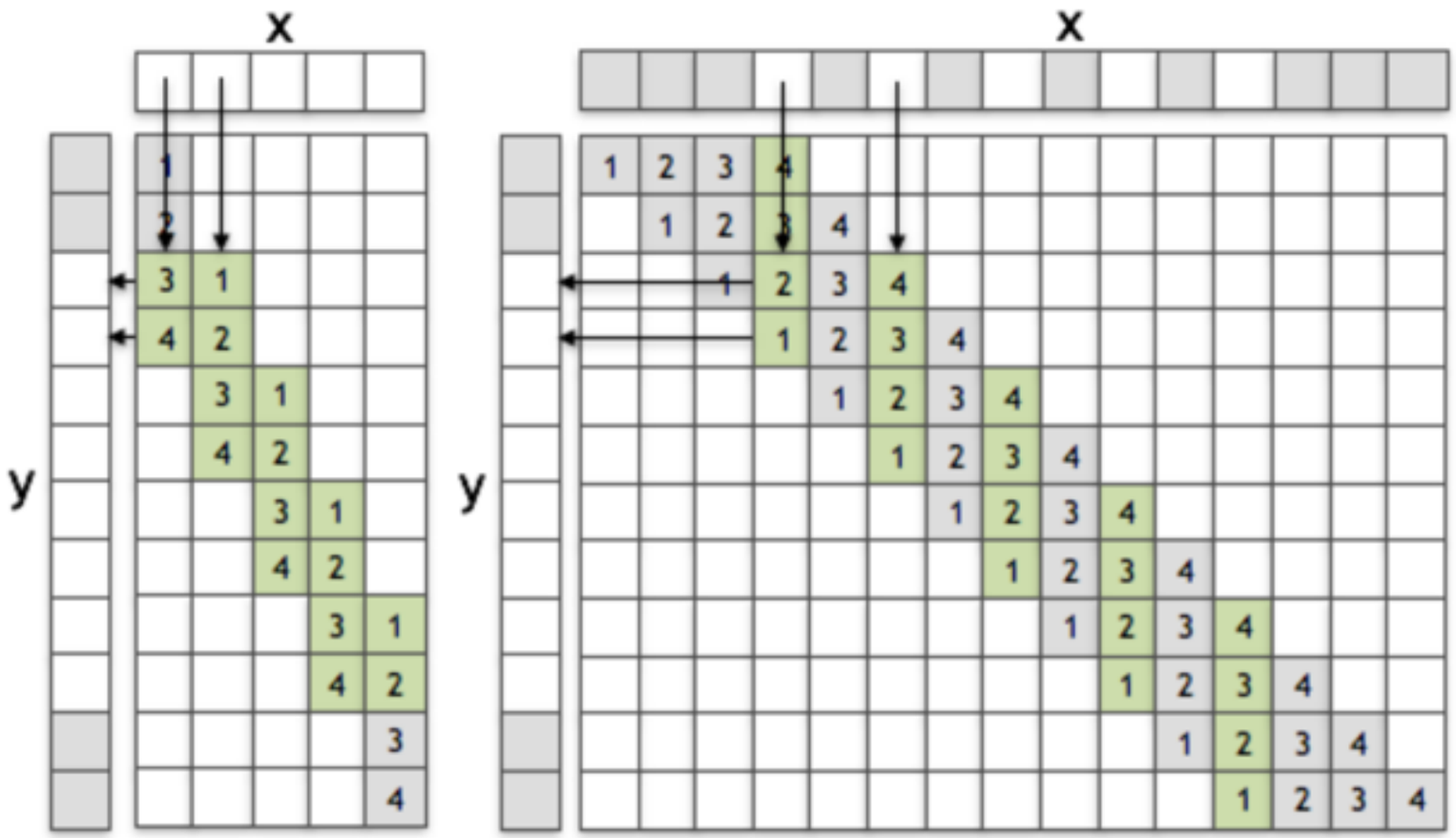


그치만 이 deconvolution을 쓰진 않았습니다.

잠깐, fractional-strided convolution?

- 기본적으로 deconvolution의 개념으로, convolution의 inverse같은 어프로치 중 하나. Transposed convolution이라고도 함.

- 근데 웹에 공개된 코드들에서는 어떤 경우 tensorflow의 deconv 그냥 쓰는듯.. tf.nn.conv2d_transpose 가 정확한 듯.



둘이 같다는 얘기.
fractionally strided conv = transposed conv

Is the deconvolution layer the same as a convolutional layer?, Wenzhe Shi, et al.

Details of training

LSUN / Imagenet-1k / 직접 만든 Faces dataset, 이렇게 세가지 데이터에 대해서 학습.

- 이미지들에 대해 tanh activation의 range인 [-1, 1]로 스케일링했다. 그 외에는 pre-processing 작업은 하지 않았다.

- 모든 모델은 mini-batch (128개씩) SGD(stochastic grandient descent)으로 학습하였다.

- 모든 weight값들은 Normal distribution (zero-centered/sd=0.02)으로 초기화하였다.

- Leaky ReLU의 leak 부분의 slope은 0.2로 설정하였다.

- 기존 GAN은 momentum을 사용해서 학습속도를 높였지만, 우리는 Adam optimizer를 사용하였다.

- Learning rate는 기존에 사용되던 0.001이 너무 높았기 때문에 0.0002로 사용하였다.

- Momentum term β1를 0.9 값으로 두는 것이 oscillation과 instability를 초래한다는 것을 발견했고, 0.5로 낮추는 것이 더 나은 학습을 가능하게 했다.

De-duplication issue

- Generator가 input example을 memorize할 likelihood를 더 감소시키기 위해, 우리는 간단한 image de-duplication process를 적용했다.

- 우리는 3072-128-3072 de-noising dropout regularized ReLU autoencoder를 32 x 32 downsampled center-crop 부분들에 적용했다. (즉, **center crop부분을 unsupervised learning으로 generate해서 어느정도 generalize 했다는 뜻.** 그건 그렇고 이게 간단한 거냐..)

- 그리고 이 결과값인 code layer activation에 대해 (ReLU activation에) threshold를 둠으로써 이진화했다. : 이 방식은 information을 보존하는 효과적인 방법 (*Understanding Locally Competitive Networks, https://arxiv.org/abs/1410.1165*) 이고, semantic-hashing을 하는 (code 부분에 대해) 편리한 form을 제공함으로써 linear time으로 de-duplication을 가능하게 해준다.

- Hash collision에 대한 시각적 관찰을 해보면 100개중 1개 정도의 false positive를 보이는 높은 precision을 볼 수 있다.

- 또한 이 방법은 약 275,000개의 duplicate를 발견해서 제거하였고, 이는 높은 recall값을 보여준다고 할 수 있다.

Validation - CIFAR-10 dataset

- Unsupervised learning의 퀄리티를 측정하는 일반적인 방법은 이를 feature extractor로 사용해서 supervised learning의 결과값을 평가해보는 것.

- CIFAR-10 dataset 문제에 대해 제일 기본이 되는 방법은 K-means 방법으로 feature를 뽑아내는 single layer를 사용하는 방법이다. 4800개 정도의 큰 feature map을 사용하면 80.6%의 정확도를 얻을 수 있다. Multi-layer로 feature extraction을 수행하면 82.0%까지 올릴 수 있다.

- DCGAN으로 학습한 representation들의 퀄리티를 평가하기 위해, 우리는 Imagenet 1-k에 대해 학습시킨 후에 discriminator의 모든 레이어에서의 convolutional feature를 사용했다. 각 레이어의 representation을 max-pooling해서 4 x 4 spatial grid를 뽑아냈다. 이 feature들을 flatten / concatenate해서 28672 dimension의 벡터를 만들었고, regularized linear L2-SVM 모델을 그 위에다가 적용했다. 이 모델은 82.8%의 정확도를 보임으로써 K-means방식을 사용한 모델들을 모두 이겼다.

- 주목할 만한 것은, discriminator는 훨씬 적은 수의 feature map을 사용했다는 것인데(가장 많은 레이어가 512개), 사실 총 feature vector 크기는 훨씬 커진다. (4 x 4 spatial 영역을 보는 여러개의 레이어로 이루어져 있으므로)

- DCGAN을 사용한 방법은 사실 Exemplar CNN보다는 더 낮은 결과치를 보여준다. (Discriminative unsupervised feature learning with exemplar convolutional neural networks) 이 테크닉은 평범한 discriminator CNN을 unsupervised 방식으로 학습시켜서 exemplar sample을(특별히 골라서 매우 augment시킨 sample들) source dataset과 구분하는 방법이다.

- Discriminator의 representation들을 finetuning함으로써 더 나은 결과를 얻을 수도 있지만 나중에 해보겠다. (..)

- 추가적으로 우리 DCGAN 모델은 CIFAR-10 dataset으로 학습시킨 것이 아니기 때문에, 학습된 feature들이 domain robust 한 녀석들이라는 것도 알 수 있다.

Validation - SVHN digit classification

- Label된 데이터가 부족한 편인 SVHN (Street View House Number) 문제에도 적용해보았다. CIFAR-10때와 비슷한 방식으로 데이터를 준비했다. 10,000 example을 validation set으로 나누어서 이를 여러 hyperparameter와 model selection을 위해 사용했다.

- 1000개의 (class가 uniform하게 distribute된) 데이터를 무작위로 선택해서 regularized linear L2-SVM 모델을 적용했는데, 여기에 우리의 feature extractor pipeline을 마찬가지로 적용했다.

- 이 모델은 state of the art 성능을 보여주었다. (22.48% text error) 또다른 모델 (CNN의 변형을 사용해서 unlabeled data의 영향력을 올려주는 모델)보다 나은 성능을 보였다.

- 추가적으로 우리는 같은 architecture의 purely supervised CNN모델을 만들어서(64개의 hyperparameter set을 랜덤하게 적용해서 optimize해봄.) 비교함으로써, DCGAN에서 사용한 **CNN의 architecture**가 성능 개선의 핵심 역할을 하는 게 **아님**을 보였다. 우리 모델이 28.87%라는 훨씬 나은 성능을 보였다.

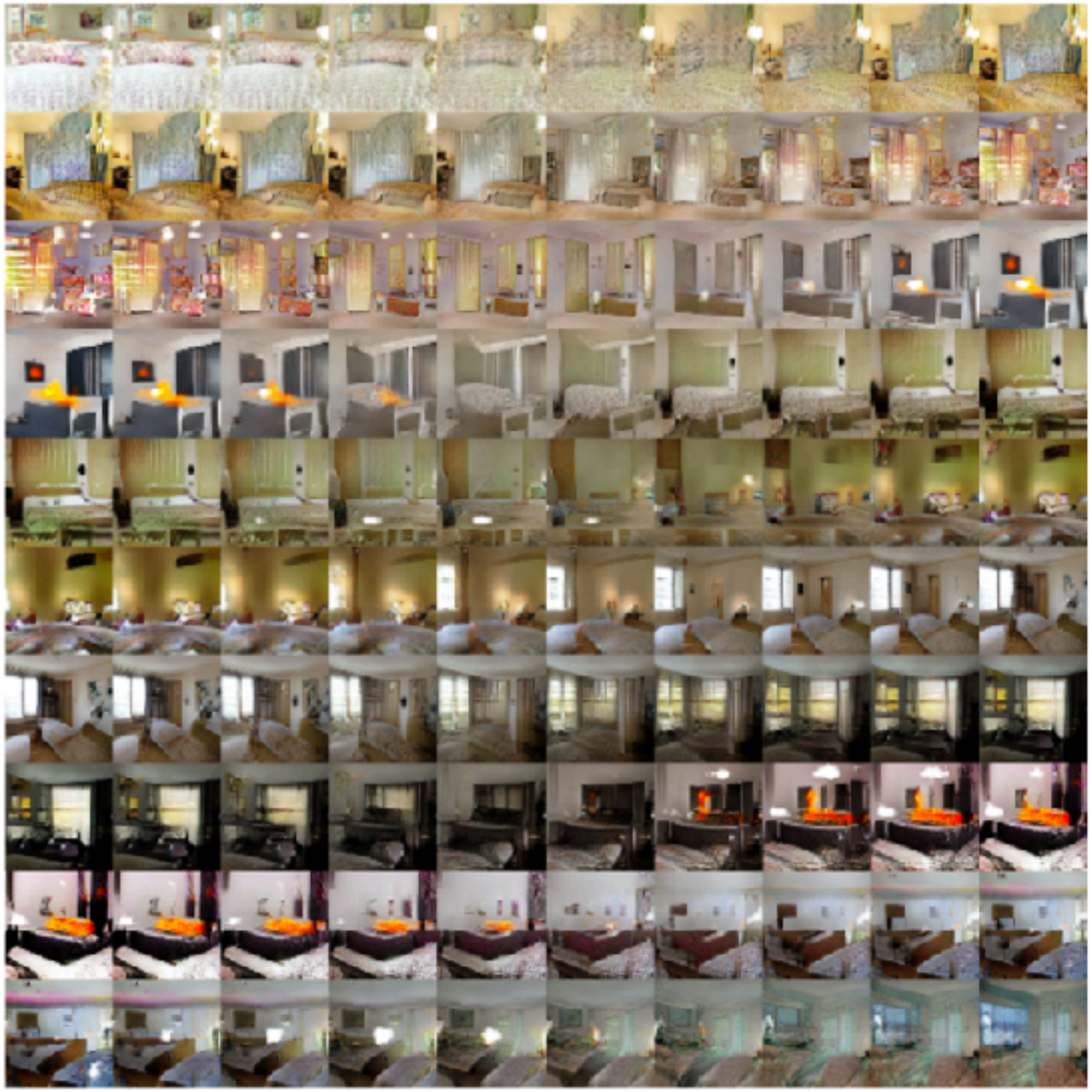
내부 시각화 - Latent space를 따라 걸어가보자.

- 먼저 우리는 latent space의 지형이 어떻게 구성되어 있는지 이해해보고자 했다.(따라 걸어보자..)

- 보통 학습된 부분에 대해 다면적으로 ‘따라 걸어보면’ memorization 여부를 알 수 있다. (sharp한 transition이 있으면 memorization이 이루어졌다는 의미) 또한 space가 계층적으로 collapsed된 방식에 대해서도 알 수 있다.

- Latent space를 따라 걷는 것이 image generation에 있어서 semantic한 변화를 보여준다면,(semantic하다는 것은 특정 object가 추가되거나 제거된다던지 하는 류의 변화) 우리는 모델이 실제적으로 이미지 생성에 관련되고 흥미로운 representation을 학습했다고 여길 수 있는 것이다.

내부 시각화 - Latent space를 따라 걸어가보자.



Latent space Z에서 랜덤한 포인트 10개를 뽑아서 그 사이에 9번의 interpolation을 수행한 결과.

이는 네트워크가 smooth한 transition을 학습했다는 것을 보여준다. (interpolation 사이의 각 이미지는 충분히 침실처럼 보인다.)

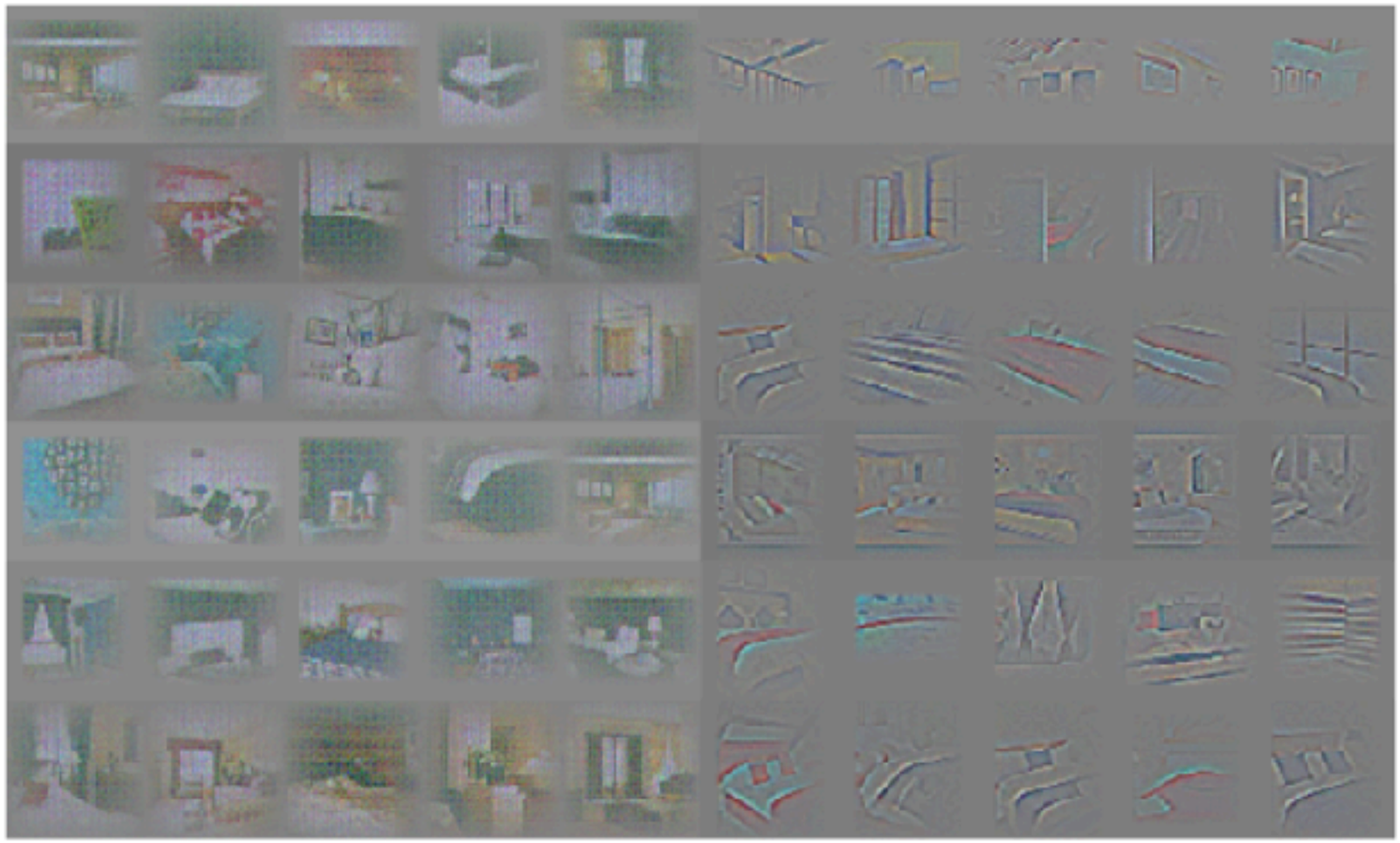
내부 시각화 - Discriminator의 feature에 대한 시각화.

- *Visualizing and Understanding Convolutional Networks (https://arxiv.org/abs/1311.2901)* 논문을 보면, 큰 이미지 데이터셋에 대해 학습한 (unsupervised learning) CNN이 매우 강력한 feature를 보여줌을 알 수 있다.

- 또한, scene classification에 대해 학습한(supervised) CNN은 object detector를 학습했음을 알 수 있다. *(Learning and transferring mid-level image representations using convolutional neural networks, http://lear.inrialpes.fr/workshop/allegro/slides/oquab.pdf)*

- 우리는 큰 이미지 데이터셋에 대해 unsupervised 학습한 DCGAN 또한 매우 흥미로운 feature의 계층구조를 학습했음을 발견했다. Guided backpropagation을 사용해서 우리는 discriminator를 시각화했는데, 침실의 특정 부분(침대나 창문 같은)에서 activate되는 feature를 볼 수 있었다.

내부 시각화 - Discriminator의 feature에 대한 시각화.



Random filters　　Trained filters

Guided backpropagation을 사용한 시각화. : Discriminator의 마지막 레이어에서 첫 6개의 convolutional feature에 대해 maximal axis-aligned response를 시각화한 것.

> LSUN 침실 데이터셋에서 중심이 되는 오브젝트인 침대에 대해서 반응하는 특정한 소수의 feature들이 있음을 볼 수 있다.
(참고로 왼쪽의 랜덤하게 초기화한 feature에서는 어떤 구분을 하기 위한 feature response를 보이지 않는다.)

Generator representation을 가지고 이상한 짓을 해보자.
: 특정 오브젝트를 그리는 것을 잊어버리도록 해보자.

- 지금까지 discriminator가 학습한 representation을 봤다. 그럼 generator는 어떤 것을 학습했을까. Generate된 샘플의 퀄리티를 보면 generator가 scene에서 중요한 오브젝트들(침대, 창문, 램프, 문, 가구 등)에 대한 representation을 학습했다는 것을 알 수 있다.

- 이 representation들이 학습된 구조를 파헤쳐보기 위해, 우리는 generator에서 창문을 제거하는 시도를 해보았다.

- 우리는 150 개의 샘플에서 52개의 창문에 대한 bounding box를 직접 그려넣었다.

- 그리고는 두번째로 높은 convolution 레이어 feature에서, logistic regression(창문에 대한 feature activation이 일어났는지 여부에 대한 regression)을 fit했다. : 우리가 그려넣은 bounding box 내부에서는 activation값이 positive하고 같은 이미지에서 random한 샘플들은 negative하도록 학습시켰다.

- 이런 간단한(?) 모델을 통해서, 0보다 큰 weight값들로 이루어진 feature map들은(전부 200개였다.) 모두 drop하였다. (모든 spatial location에서)

- 그리고 난 후, 새로 랜덤한 sample을 generate해보았다. (feature map을 제거한 버전 / 제거하지 않은 버전 둘다 해보았다.)

Generator representation을 가지고 이상한 짓을 해보자.
: 특정 오브젝트를 그리는 것을 잊어버리도록 해보자.

Window dropout을 적용한 / 적용하지 않은 generated image

흥미롭게도, 네트워크는 창문을 그리는 것을 잊어버리고, 그 위치에 다른 오브젝트로 채워 넣었다!

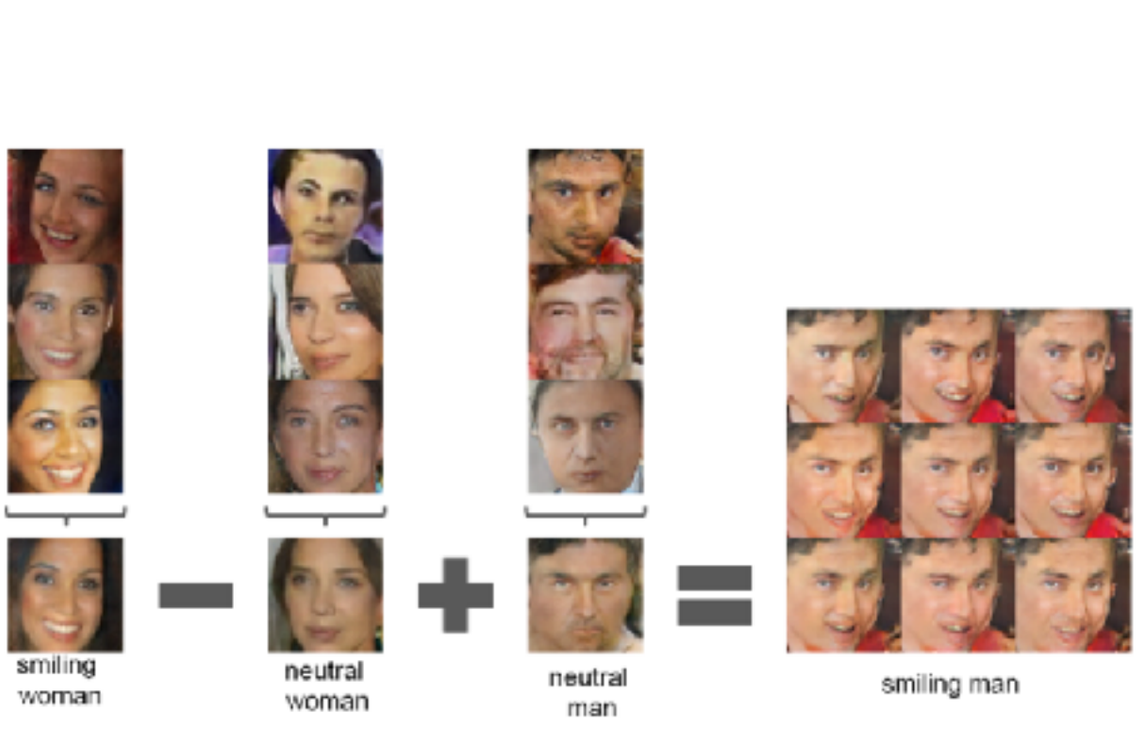
Generator representation을 가지고 이상한 짓을 해보자.
: Face sample에 대한 vector arithmetic 연산.

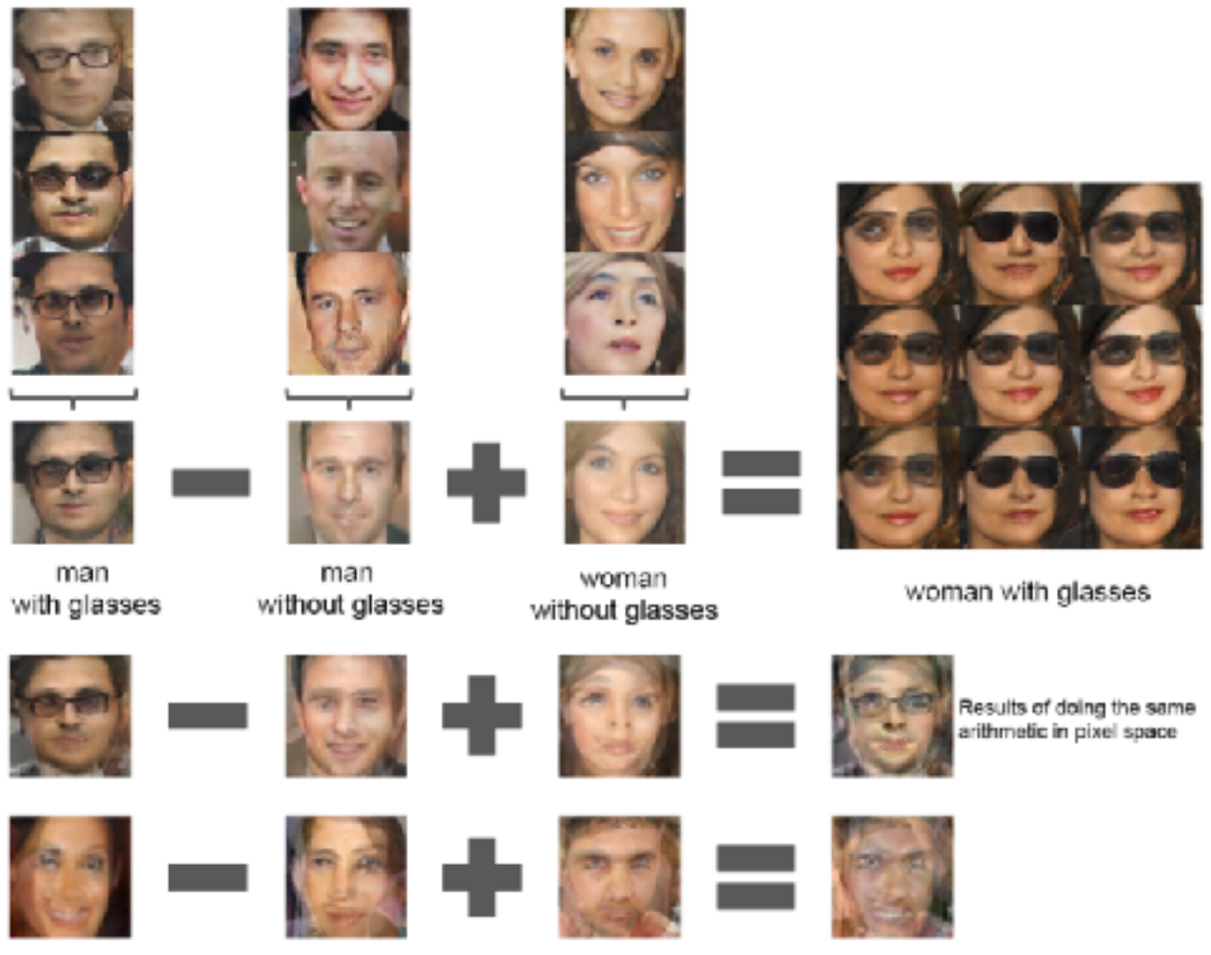
- *Distributed representations of words and phrases and their compositionality. (https://arxiv.org/abs/1310.4546)* 논문에서는 representation space에서 매우 풍부한 linear structure를 발견했다. (이 space에서 간단한 arithmetic 연산을 수행함으로써) 대표적인 예로, vectore("King")에서 vector("Man")을 뺀 후 vector("Woman")을 더한 값은 nearest neighbor가 vector("Queen")이 되는 것을 보였다.

- 우리는 이런 structure가 우리의 generator의 Z representation에서도 보이는지 확인해보았다. 우리는 비슷한 arithmetic 연산을 특정 visual concept에서의 대표적인 샘플들에서 수행해보았다.

- 하나의 concept에 대해서 하나의 샘플에 대해 실험해보았을 경우에는 잘 동작하지 않았지만, 3개의 샘플에 대한 Z vector의 average값에 대해서 수행했을 때는 매우 안정적으로 결과값을 얻을 수 있었다. (arithmetic 연산에 대해 semantic한 결과값을 얻을 수 있었음.)

Generator representation을 가지고 이상한 짓을 해보자.
: Face sample에 대한 vector arithmetic 연산.





Generator representation을 가지고 이상한 짓을 해보자.
: Face sample에 대한 vector arithmetic 연산.

Face pose 또한 Z space에서 모델링 될 수 있음을 보였다.

이전에 *Learning to generate chairs with convolutional neural networks* 논문에서는 *(https://arxiv.org/abs/1411.5928)* conditional generative model이 오브젝트의 scale, rotation, position 등과 같은 attribute들을 학습할 수 있다는 것을 보였다.
하지만 우리가 아는 바에 의하면 pure unsupervised learning에서 이런 attribute를 학습할 수 있음을 보인 것은 우리가 처음이다.

결론 및 향후 과제

- 모델을 너무 오래 학습시키면, 때때로 몇몇 필터들이 하나의 oscillating mode에 빠지는 것을 볼 수 있었다. 이러한 불안정함을 해결해야 한다.

- 우리는 이 framework을 비디오(frame prediction)나 오디오(speech synthesis를 위한 pre-trained feature) 생성 모델 도메인으로 확장하는 것이 매우 흥미로울 것으로 생각한다.

- 또한 latent space의 특성에 대한 더 깊이 있는 조사도 흥미로운 연구주제일 것이다.