Deep Convolutional GANs

: Meaning of Latent Space

ISL Lab Seminar

Hansol Kang

ISL Image System Laboratory

Contents

- Review of GAN
- DCGAN
- Experiment
- Summary

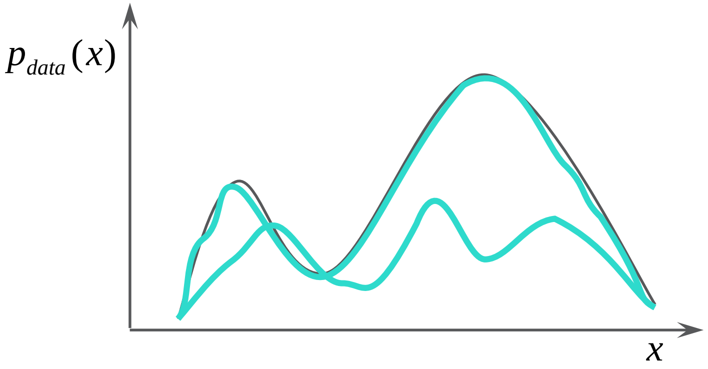
Review of GAN

- Adversarial nets

“Generative Adversarial Networks”

Goal　　　　　Method

Vs



1) Global Optimality of $p_g = p_{data}$

2) Convergence of Algorithm

DCGAN*

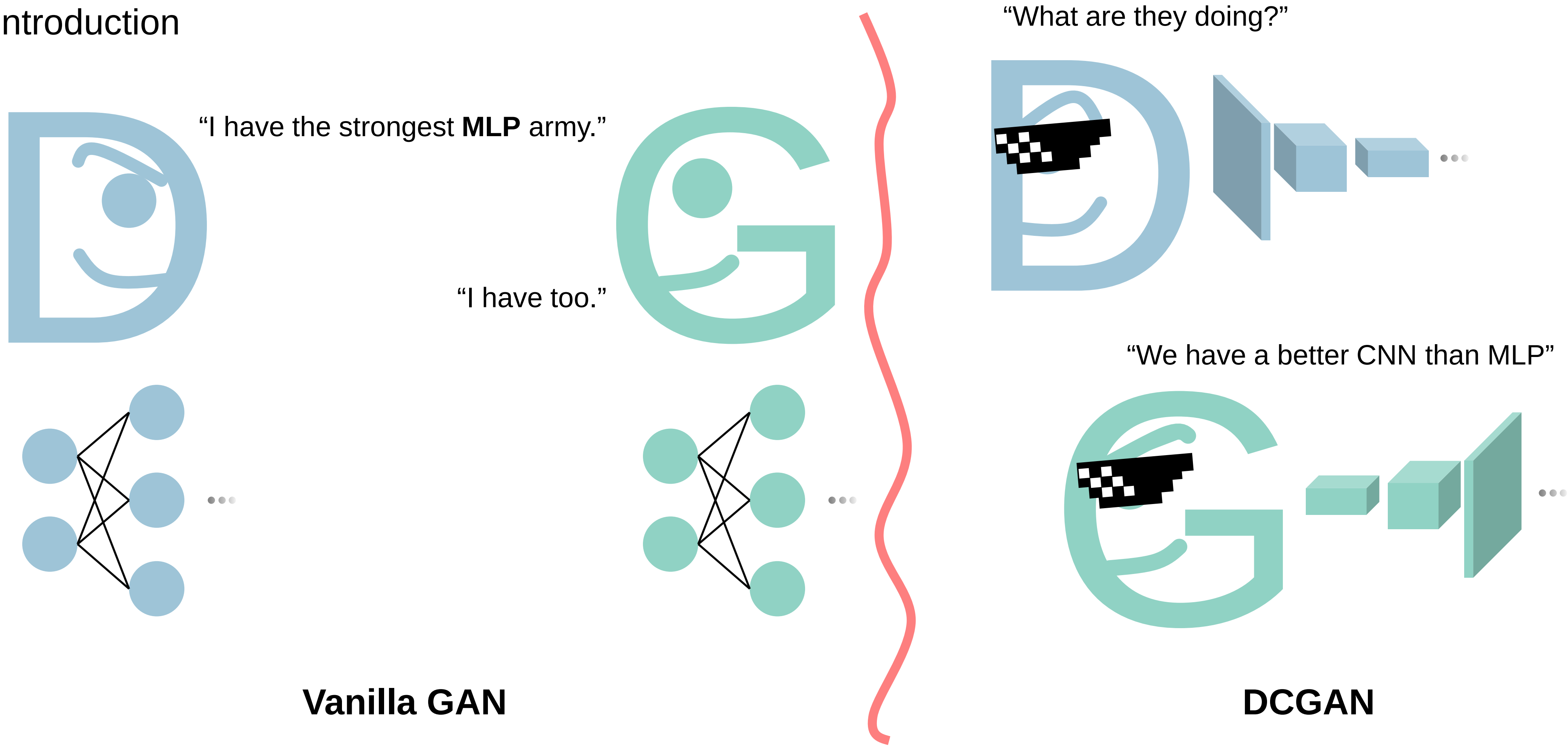
- Introduction

* *Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).*


A FEW
MOMENTS LATER

DCGAN*

- Introduction



** Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).*

DCGAN

- Contributions

Generating Natural Image

Vector arithmetic properties

Deep Convolutional GANs

Filter Visualization

Image Classification using D

DCGAN

- Contributions

Generating Natural Image

Vector arithmetic properties

Deep Convolutional GANs

Filter Visualization

Image Classification using D

DCGAN

- Approach and Model Architecture

Replace any pooling layers with **strided convolutions (discriminator)** and **fractional-strided convolutions (generator)**.

Use **batchnorm** in both the generator and the discriminator.

Remove fully connected hidden layers for deeper architectures.

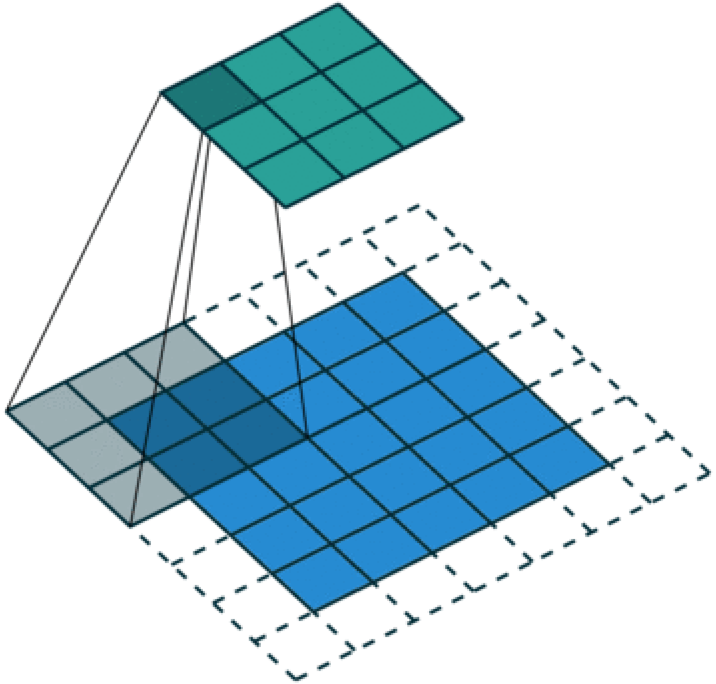
Use ReLU activation in generator for all layers except for **the output**, which **uses Tanh**.

Use LeakyReLU activation in the **discriminator** for all layers.

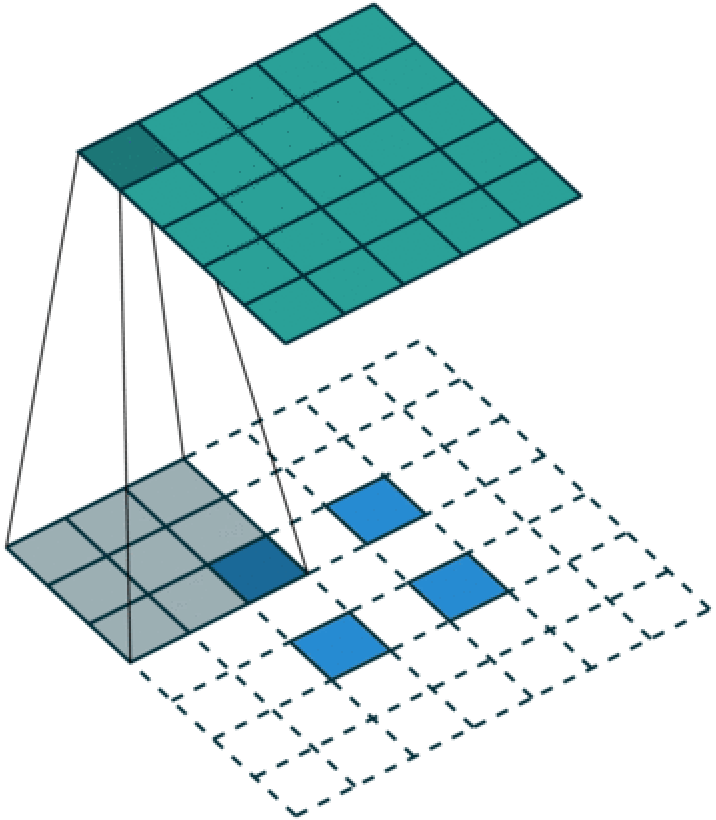
DCGAN

- Approach and Model Architecture

Strided Convolution



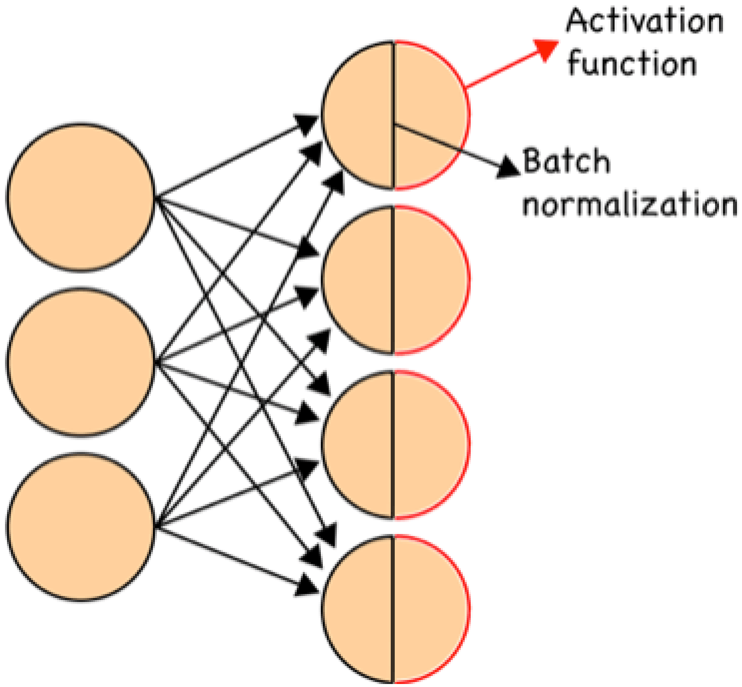
Fractional Convolution(Transposed Convolution)



DCGAN

- Approach and Model Architecture

Batch Normalization



$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

Except for these layers.

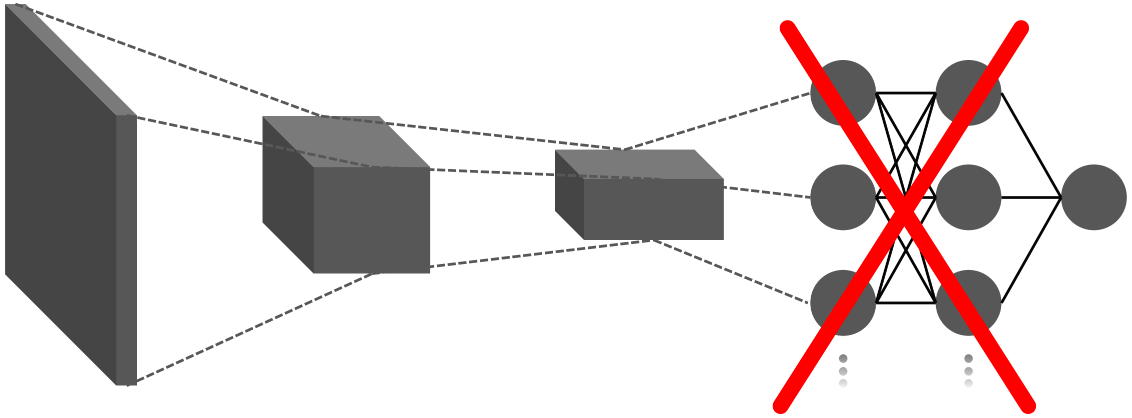
Output layer of Generator
Input layer of Discriminator

DCGAN

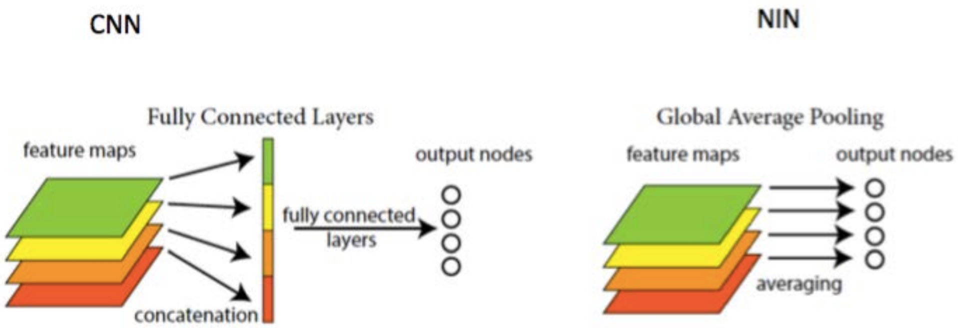
- Approach and Model Architecture

No fully connected layer

Classical CNN



GAP(Global Average Pooling)

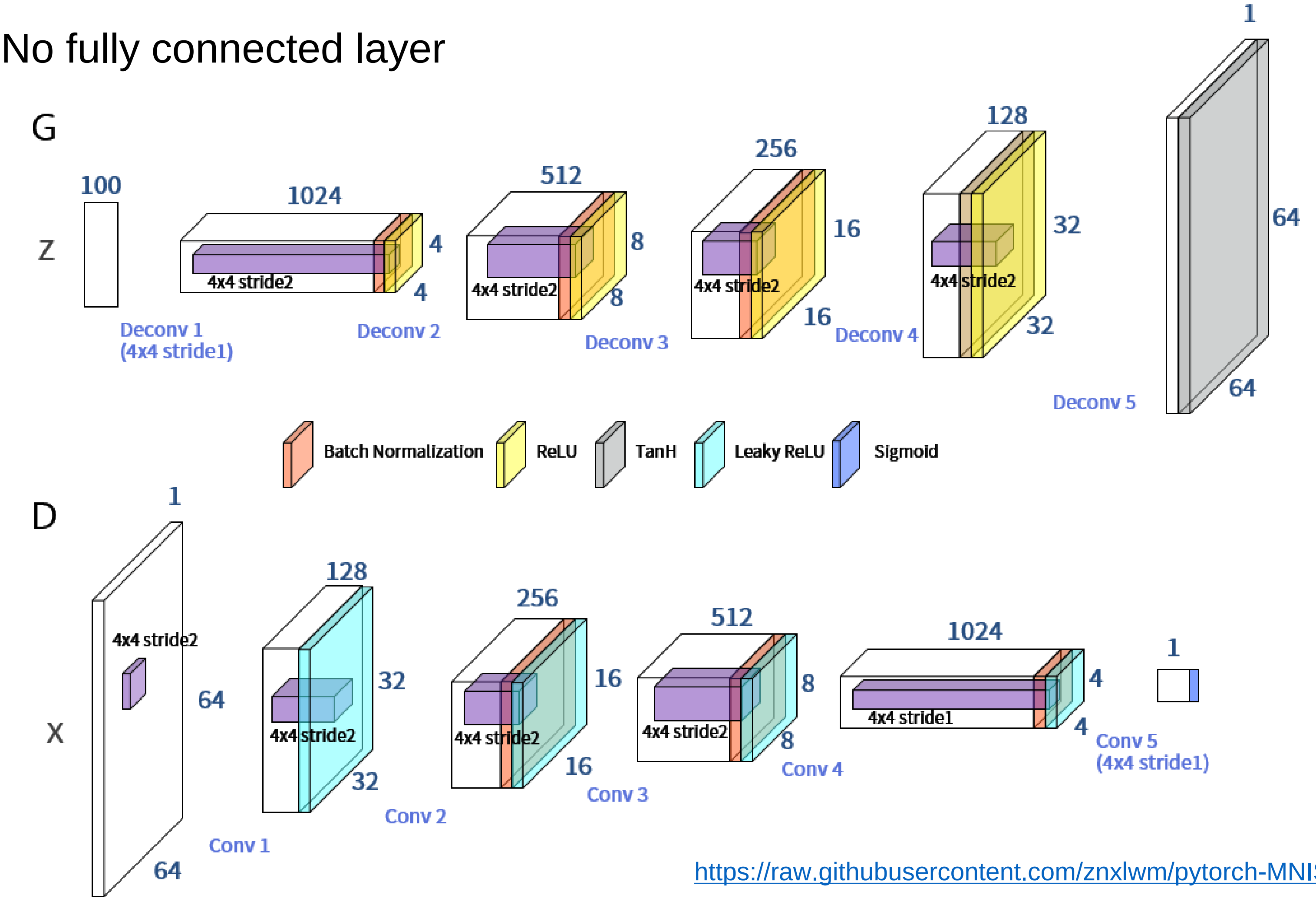


http://nmhkahn.github.io/Casestudy-CNN

DCGAN

- Approach and Model Architecture

No fully connected layer

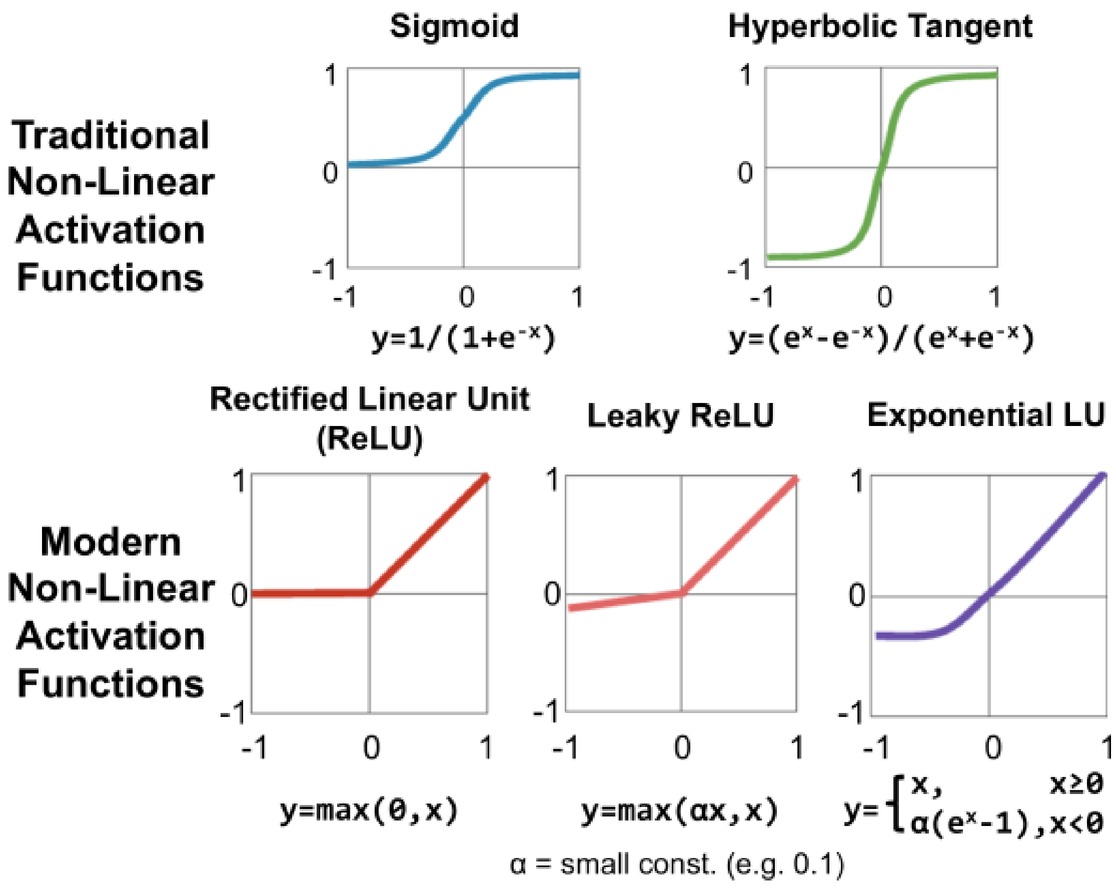


https://raw.githubusercontent.com/znxlwm/pytorch-MNIST-CelebA-GAN-DCGAN/master/pytorch_DCGAN.png

DCGAN

- Approach and Model Architecture

ReLU, Tanh, LeakyReLU



Generator : ReLU, Tanh

Discriminator : LeakyReLu , Sigmoid

http://gmelli.org/RKB/Rectified_Linear_Unit_(ReLU)_Activation_Function

DCGAN

- Details of Adversarial Training
  - Mini-batch stochastic gradient descent(SGD); mini-batch size of 128
  - All weights initialized from a zero-centered Normal distribution with standard deviation 0.02
  - Leaky slope 0.2
  - Adam optimizer; lr =0.0002, beta1 = 0.9, beta2 = 0.5

DCGAN

- Details of Adversarial Training

LSUN dataset

1 epoch

DCGAN

- Details of Adversarial Training

LSUN dataset

5 epochs

DCGAN

- Empirical Validation of DCGANs Capabilities

Table 1: CIFAR-10 classification results using our pre-trained model. Our DCGAN is not pre-trained on CIFAR-10, but on Imagenet-1k, and the features are used to classify CIFAR-10 images.

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

- CIFAR-10
- Classification
- Domain robustness

DCGAN

- Empirical Validation of DCGANs Capabilities

Table 2: SVHN classification with 1000 labels

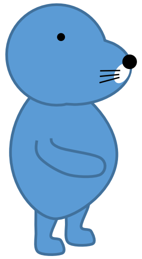
Model	error rate
KNN	77.93%
TSVM	66.55%
M1+KNN	65.63%
M1+TSVM	54.33%
M1+M2	36.02%
SWWAE without dropout	27.83%
SWWAE with dropout	23.56%
DCGAN (ours) + L2-SVM	22.48%
Supervised CNN with the same architecture	28.87% (validation)

SVHN(Street View House Numbers) dataset

DCGAN

- Investigating and Visualizing The Internals of The Networks

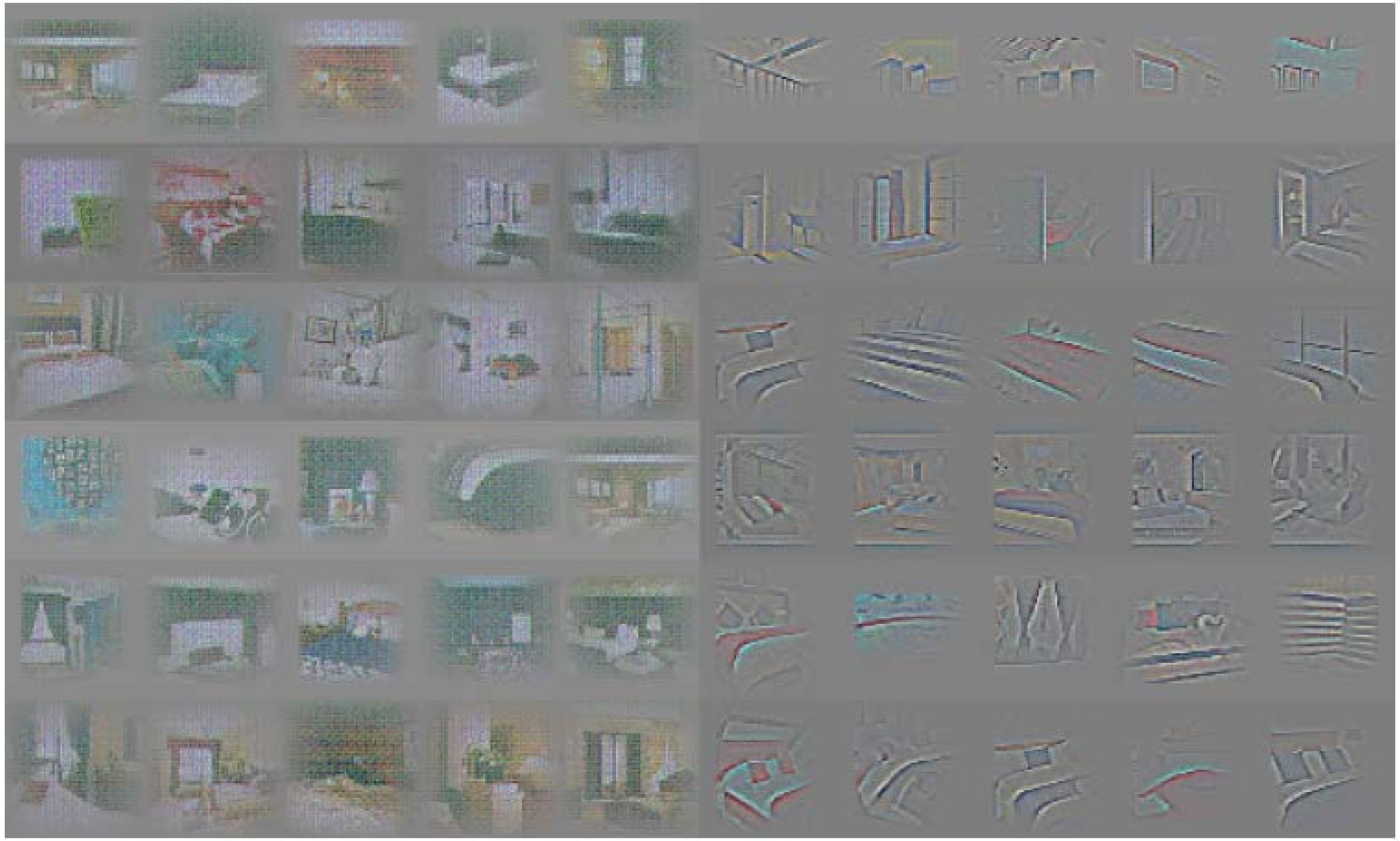
Walking in the latent space



DCGAN

- Investigating and Visualizing The Internals of The Networks(cont.)

Visualizing the discriminator features

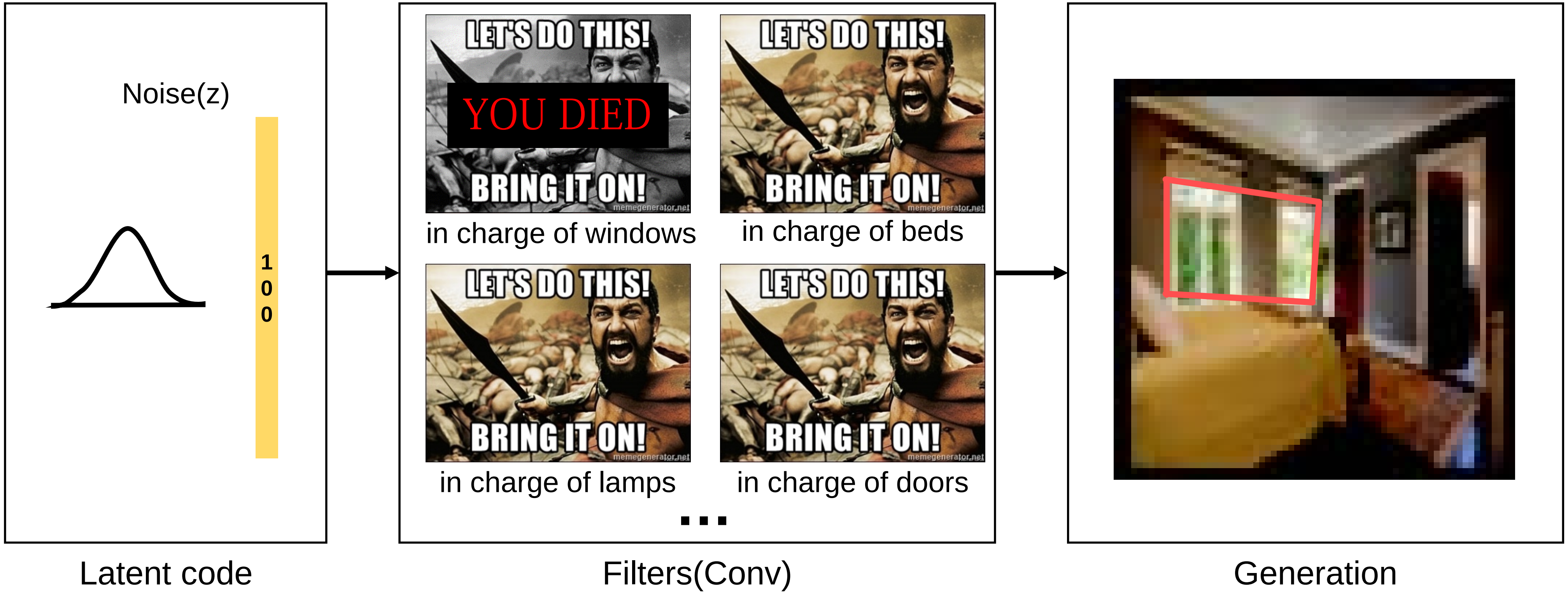


Random filters Trained filters

DCGAN

- Investigating and Visualizing The Internals of The Networks(cont.)

Forgetting to draw certain objects



DCGAN

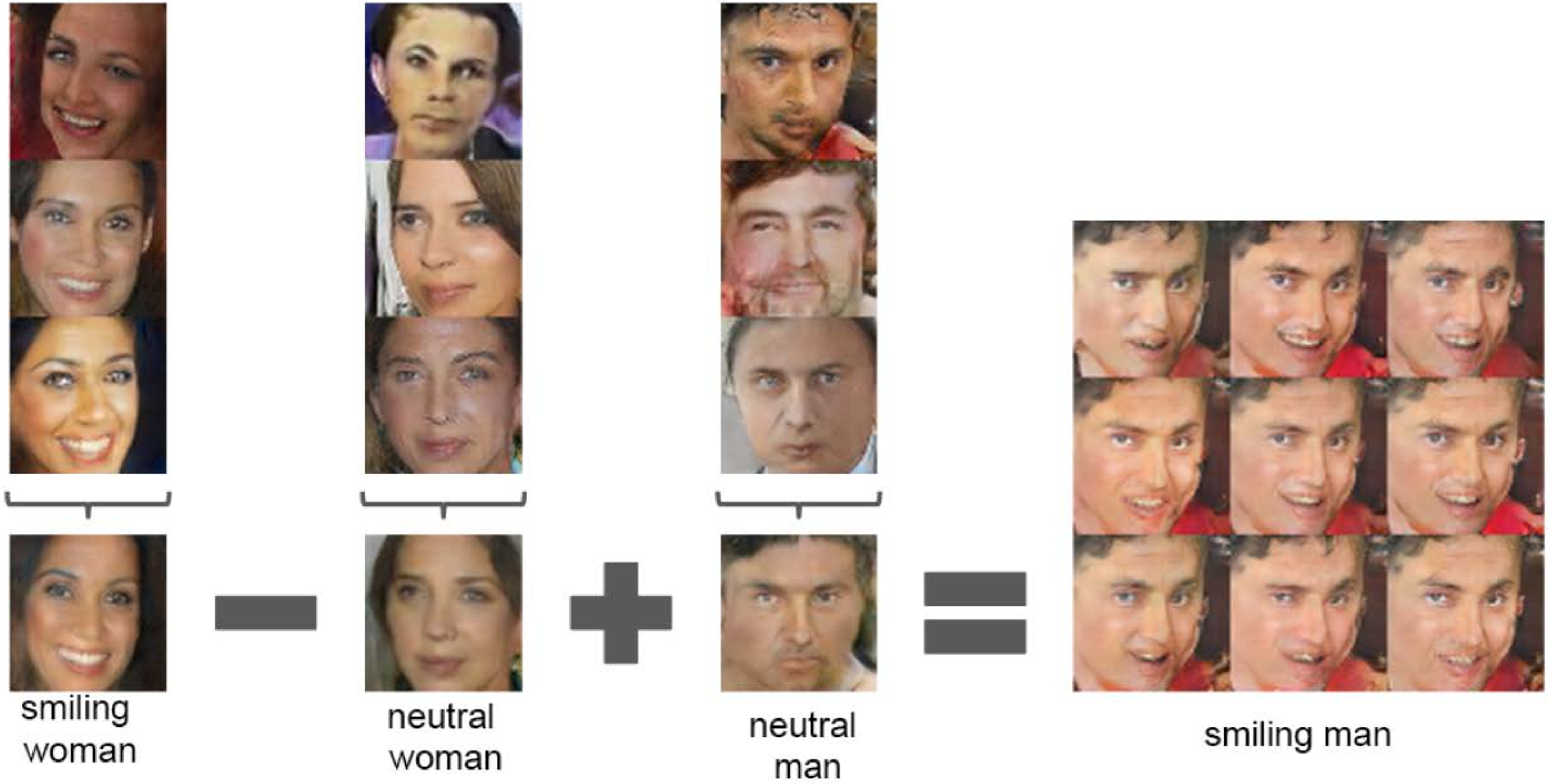
- Investigating and Visualizing The Internals of The Networks(cont.)

Forgetting to draw certain objects

DCGAN

- Investigating and Visualizing The Internals of The Networks(cont.)

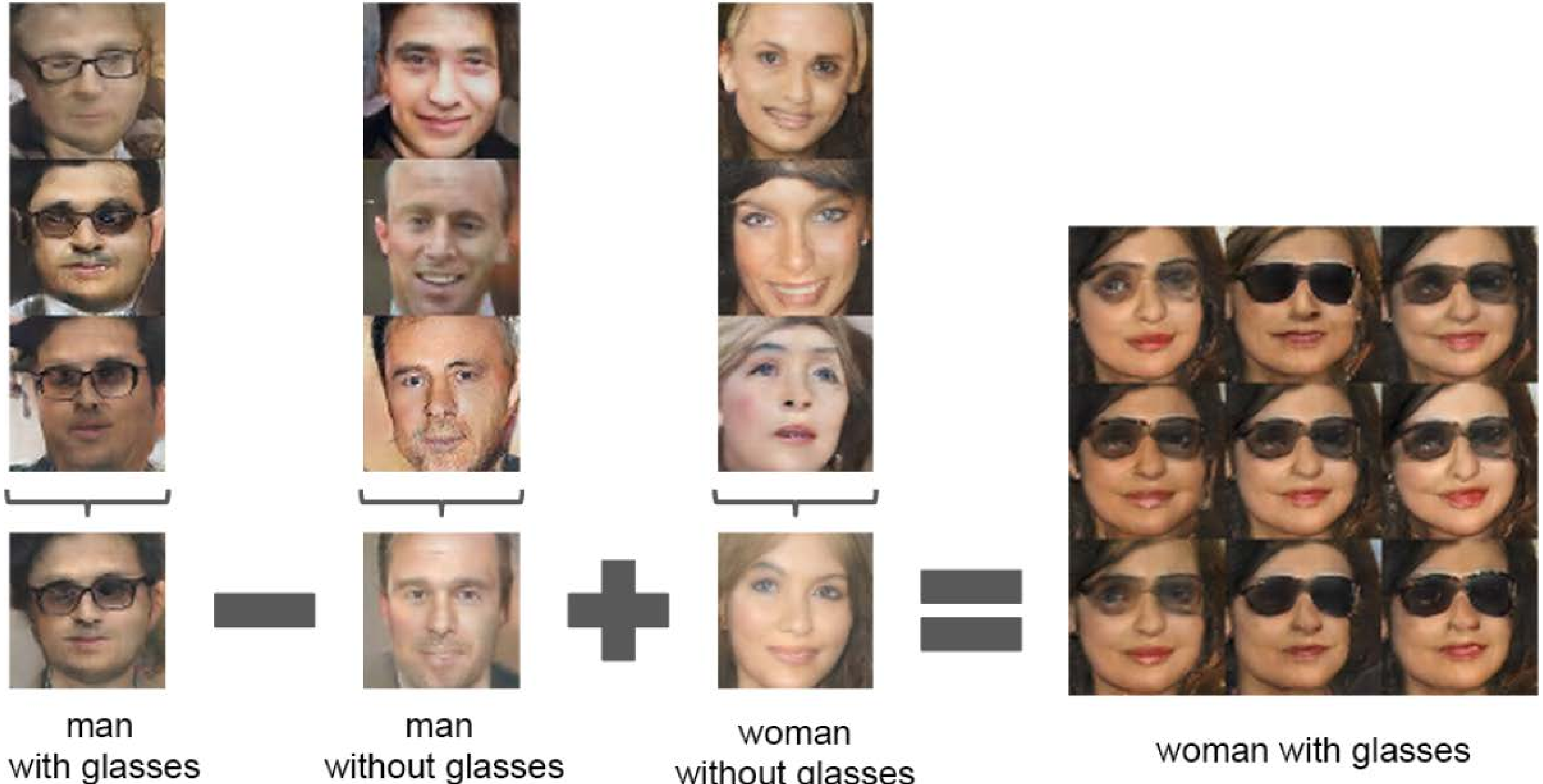
Vector arithmetic on face samples



DCGAN

- Investigating and Visualizing The Internals of The Networks(cont.)

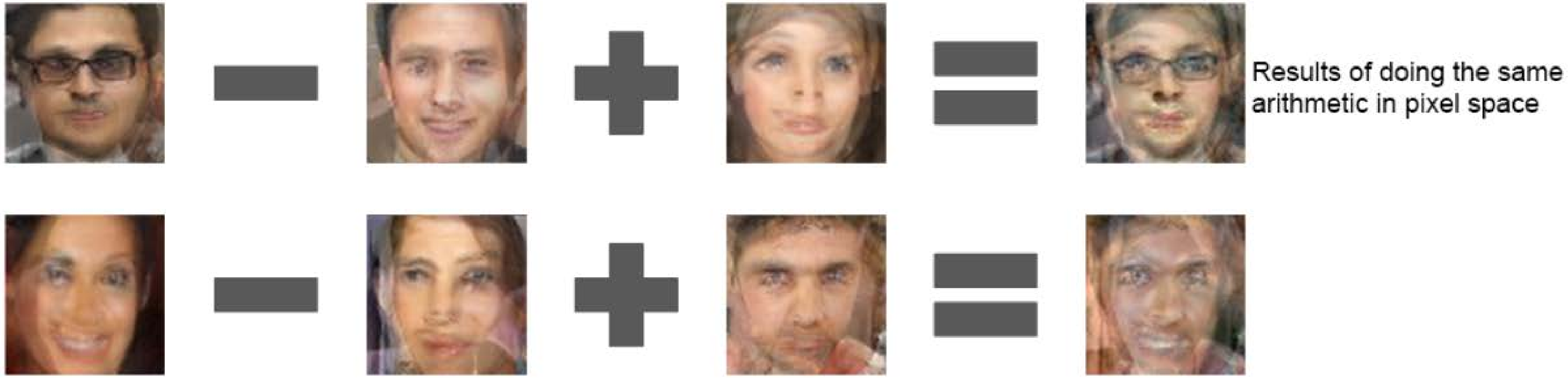
Vector arithmetic on face samples



DCGAN

- Investigating and Visualizing The Internals of The Networks(cont.)

Vector arithmetic on face samples



DCGAN

- Investigating and Visualizing The Internals of The Networks(cont.)

Vector arithmetic on face samples

Experiment

- Code

GAN_PyTorch

Various GAN implementations based on PyTorch. This project is consist of simple and standard
a relatively short code length, and only simple functions are implemented.
The Standard version has various functions rather than the simple version. It also provides a UI
standard version is loaded and executed). ~~In fact, I don't know if UI is comfortable...~~

Implementation list

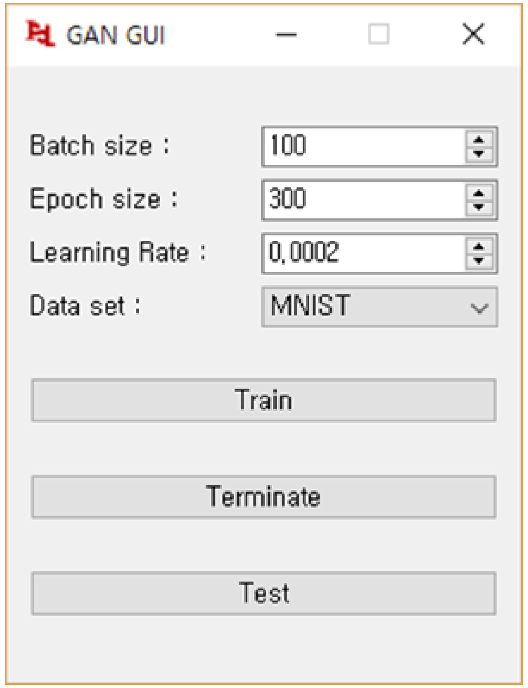
- Vanilla GAN : Simple | Standard & UI
- DCGAN : Simple |
- InfoGAN : Simple |

Experiment Environment

- Windows 10 Enterprise
- Intel i7-3770k
- RAM 12.0 GB
- NVIIDA GTX TITAN
- Python 3.6.4
- PyTorch 0.4.0
- torchvision 0.2.1

Vanilla_Standard.py and for_UI.py

- The UI supports batch size, epoch size, learning rate, and dataset settings.
- Save the log file as csv.



https://github.com/messy-snail/GAN_PyTorch

Experiment

- Code

```python
class Generator(nn.Module):
    def __init__(self, latent_sz):
        super(Generator, self).__init__()
        self.tconv1 = nn.ConvTranspose2d(latent_sz, 1024, 4, 1, 0)
        self.tconv2 = nn.ConvTranspose2d(1024, 512, 4, 2, 1)
        self.tconv3 = nn.ConvTranspose2d(512, 256, 4, 2, 1)
        self.tconv4 = nn.ConvTranspose2d(256, 128, 4, 2, 1)
        self.tconv5 = nn.ConvTranspose2d(128, 3, 4, 2, 1)

        self.bn1 = nn.BatchNorm2d(1024)
        self.bn2 = nn.BatchNorm2d(512)
        self.bn3 = nn.BatchNorm2d(256)
        self.bn4 = nn.BatchNorm2d(128)

    def forward(self, input):
        x = F.relu(self.bn1(self.tconv1(input)))
        x = F.relu(self.bn2(self.tconv2(x)))
        x = F.relu(self.bn3(self.tconv3(x)))
        x = F.relu(self.bn4(self.tconv4(x)))
        x = F.tanh(self.tconv5(x))

        return x
```

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.conv1 = nn.Conv2d(3, 128, 4, 2, 1)
        self.conv2 = nn.Conv2d(128, 256, 4, 2, 1)
        self.conv3 = nn.Conv2d(256, 512, 4, 2, 1)
        self.conv4 = nn.Conv2d(512, 1024, 4, 2, 1)
        self.conv5 = nn.Conv2d(1024, 1, 4, 1, 0)

        self.bn2 = nn.BatchNorm2d(256)
        self.bn3 = nn.BatchNorm2d(512)
        self.bn4 = nn.BatchNorm2d(1024)

    def forward(self, input):
        x = F.leaky_relu(self.conv1(input), 0.2)
        x = F.leaky_relu(self.bn2(self.conv2(x)), 0.2)
        x = F.leaky_relu(self.bn3(self.conv3(x)), 0.2)
        x = F.leaky_relu(self.bn4(self.conv4(x)), 0.2)
        x = F.sigmoid(self.conv5(x))

        return x
```

https://github.com/messy-snail/GAN_PyTorch

Experiment

- Results#1 CelebA

Still have this sample

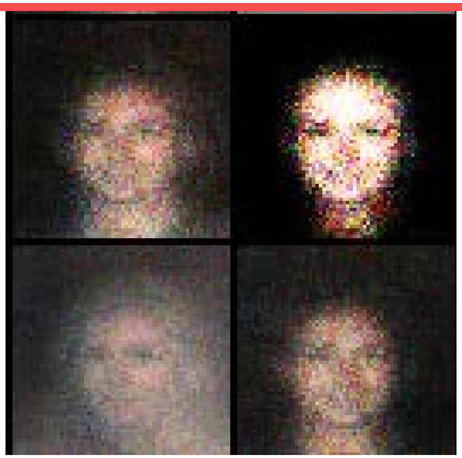
Ground Truth

Results are cherry picked

Vanilla GAN :

DCGAN :

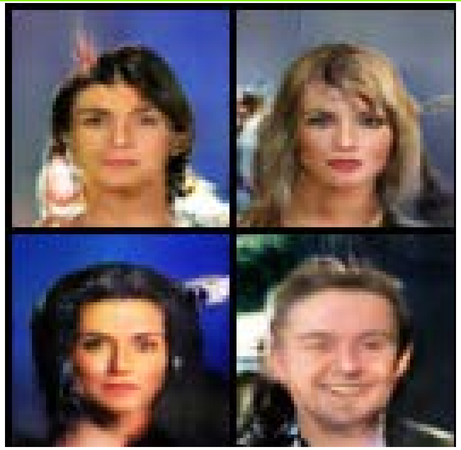
Epoch 1



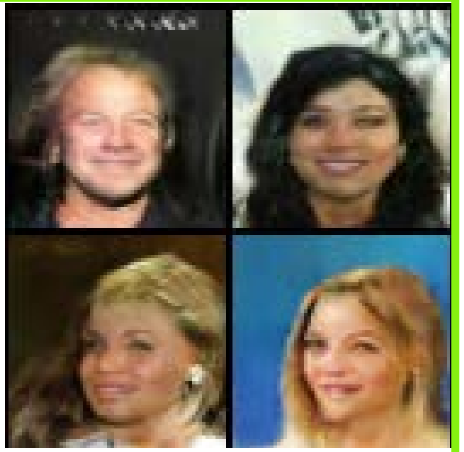
Epoch 5

Epoch 100

Epoch 1



Epoch 5



Epoch 30

Experiment

- Results#2 LSUN)

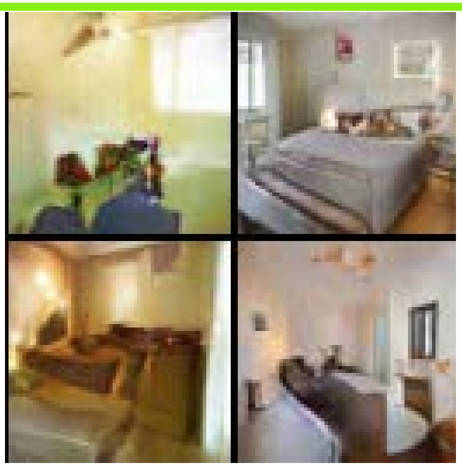
Ground Truth

Epoch 1

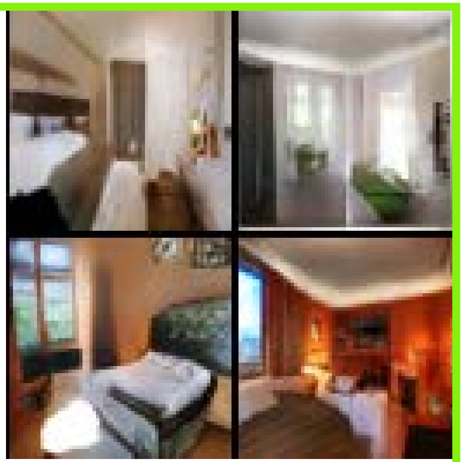
Epoch 5

Epoch 12

Epoch 1



Epoch 2



Epoch 5

Results are cherry picked

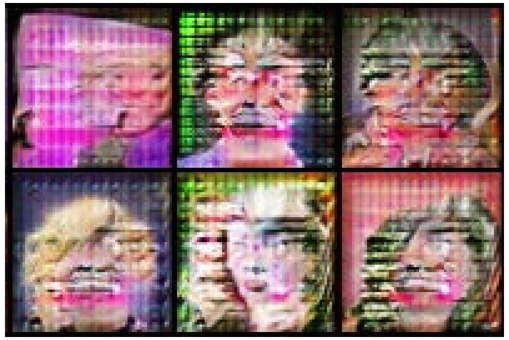
Vanilla GAN :

DCGAN :

Experiment

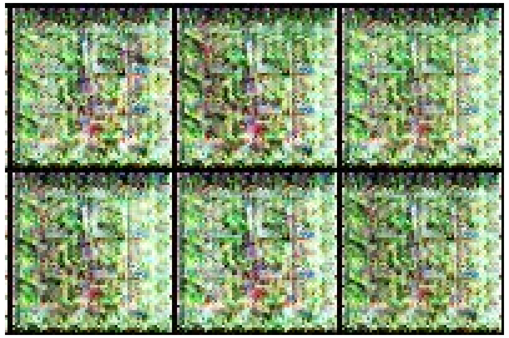
- Results#3 Korean Idol – Transfer trial

Ground Truth



Epoch 1

Epoch 2



Epoch 3

- I used weights and biases generated by celebA learning.
- I wanted the effect of transfer learning but failed.

Maybe these factors (Asian, cropping image)

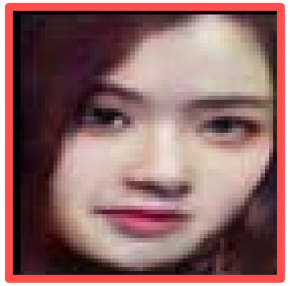
Epoch 4

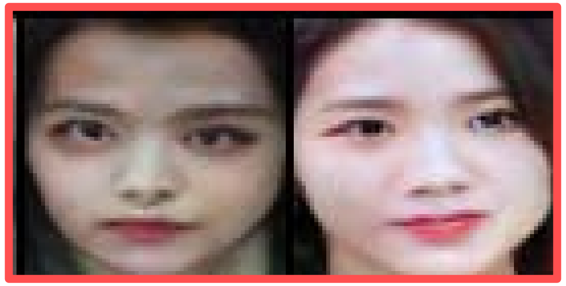
Epoch 5

Epoch 6

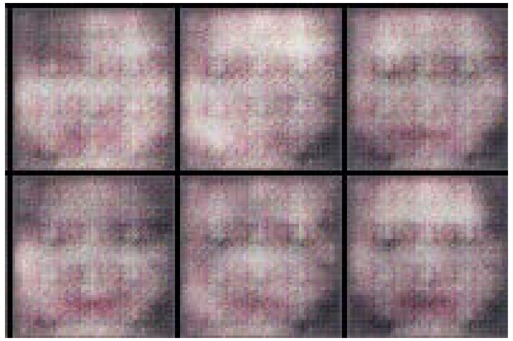
Experiment

- Results#4 Korean Idol





Ground Truth



Epoch 1

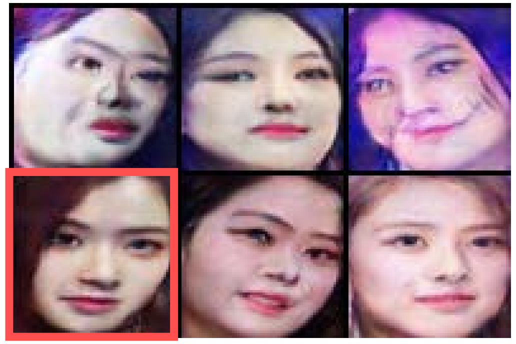
Epoch 5

Epoch 30

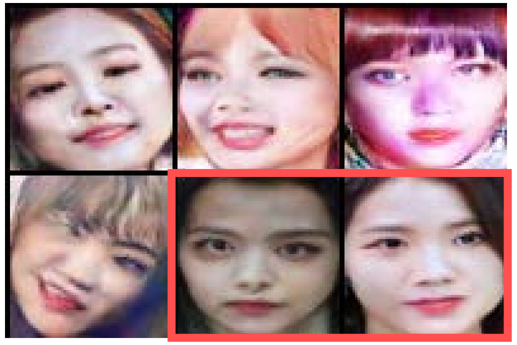
- 10000 images

Insufficient data set

Epoch 50



Epoch 100

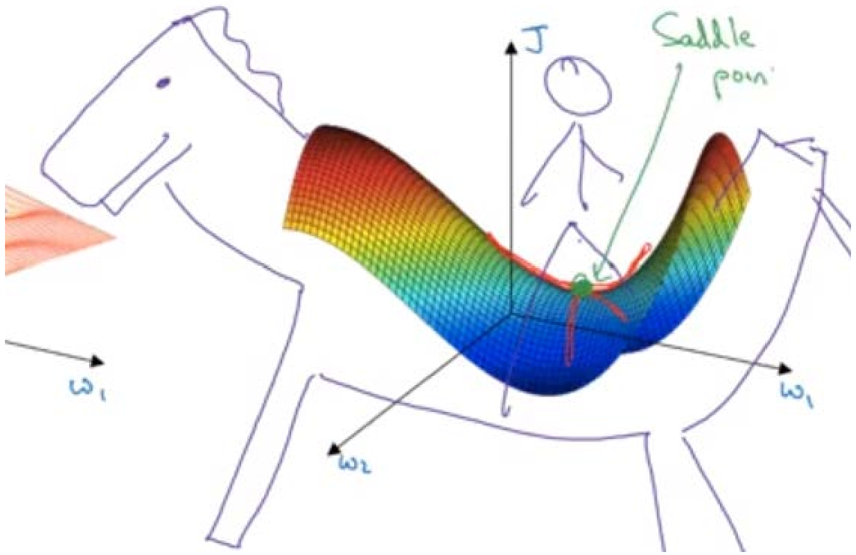


Epoch 150

Summary

- Stable set of architectures for training generative adversarial networks
- Good representations of images for supervised learning and generative modeling
- Sometimes collapse a subset of filters to a single oscillating mode
- Latent code has a special meaning, not a simple noise component.

[Instability of GAN]



Future work

Paper Review

- [x] ~~Vanilla GAN~~
- [x] ~~DCGAN~~
- [ ] InfoGAN
- [ ] Unrolled GAN
- [ ] Wasserstein GAN
- [ ] LS GAN
- [ ] BEGAN
- [ ] ~~Pix2Pix~~
- [ ] Cycle GAN

Proposed Model

- [ ] SpyGAN (working title)

Tips

- [x] ~~Document~~
- [x] ~~Programming~~

Mathematical Study

- [ ] Information theory

Q & A

Thank you for your attention

Appendix

- Issues at the VAE Seminar (18.07.23)

  - ✓ **Issue#1** Performance of VAE and GAN
  - ✓ **Issue#2** Log likelihood
  - ✓ **Issue#3** Dimension of latent code
  - ✓ **Issue#4** Why manifold?

Durk Kingma

Machine Learning researcher at OpenAI

1. Adam: A Method for Stochastic Optimization
2. Auto-Encoding Variational Bayes
- Mathematically very difficult papers

Intuitive explanation
: I refer to this video
오토인코더의 모든 것
https://www.youtube.com/watch?v=o_peo6U7IRM

Appendix

- Issue #1 Performance of VAE and GAN

"Compared to GAN, **VAE is relatively blurred** and I do not know why."

"Cost function"

VAE $\min L(\phi, \theta, x)$

$$L(\phi, \theta, x) = -E_{q_\phi(z|x)}\left[\log\left(x \mid g_\theta(z)\right)\right] + KL\left(q_\phi(z \mid x) \,\|\, p(z)\right)$$

Reconstruction Error + **Regularization**

GAN $\min_G \max_D V(G, D)$

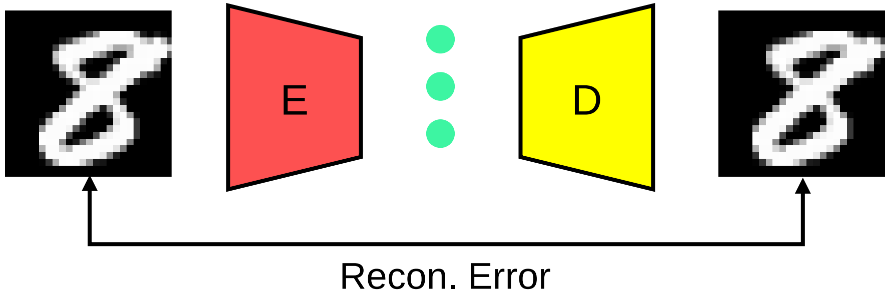
$$V(G, D) = E_{x \sim p_{data}}\left[\log D(x)\right] + E_{z \sim p_z}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

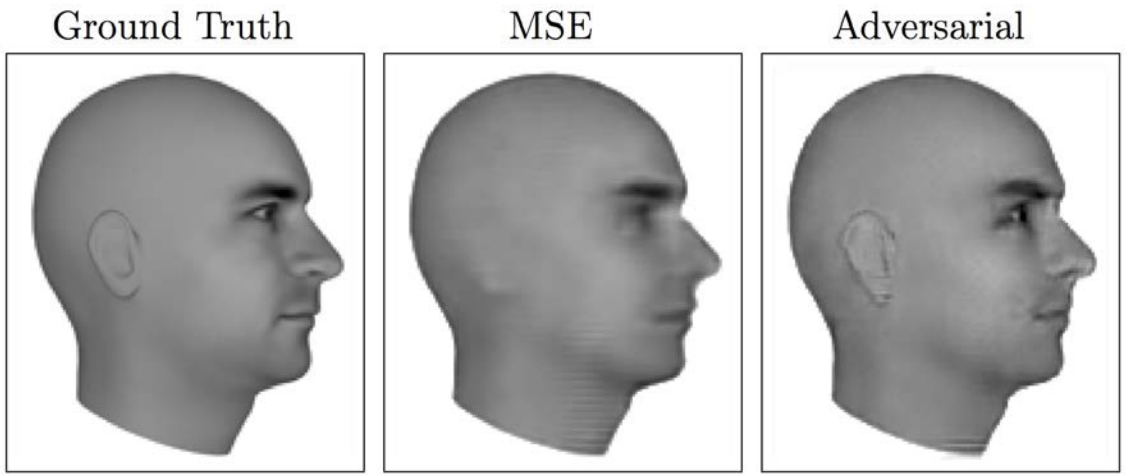
≈ D Loss + **≈ G Loss**

Appendix

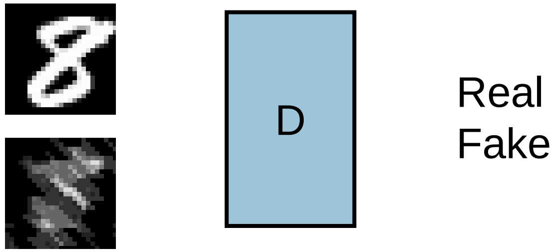
- Issue #1 Performance of VAE and GAN

VAE Loss= Recon. Error + Regularization





GAN Loss= G_Loss + D_Loss



VAE vs. GAN

1. Optimize
2. Image Quality
3. Generalization

Appendix

- Issue #2 Log likelihood

Question about log likelihood

“Summation and monotonically increasing”

eg. Gaussian Distribution
Mean and **Std**

MLE(Maximum Likelihood Estimation) : Unknown parameter estimation from observation

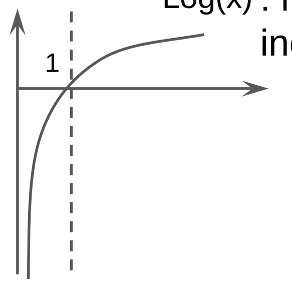
$$\hat{\theta} = \arg\max_{\theta} p(\mathbf{y} \mid \theta)$$

$$= \arg\max_{\theta} \prod_i p(y_i \mid \theta) \longrightarrow \arg\max_{\theta} \log\left(\prod_i p(y_i \mid \theta)\right) = \arg\max_{\theta} \sum_i \log p(y_i \mid \theta)$$

$\downarrow$ Generation model

$$\arg\max_{\theta} \sum_i \log p_{\theta}(x_i)$$

cf.



Log(x) : monotonically increasing function

Appendix

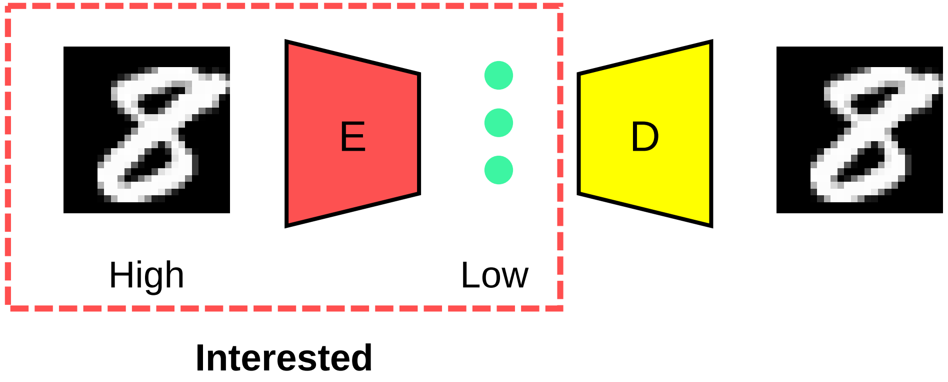
- Issue #3 Dimension of latent code

"Is the latent code dimension always small?"

"Yes"

Sparse AE

AE, What's this? Dimension reduction



Appendix

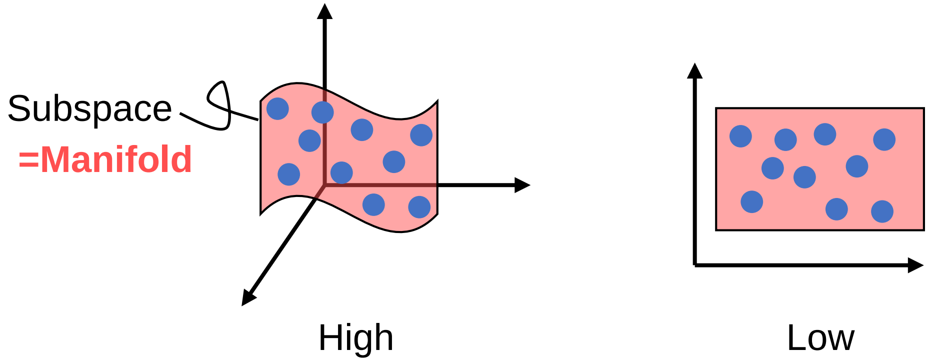
- Issue #4 Why manifold?

: Correlation between generation and manifold…

What's the manifold and Why explain the manifold?

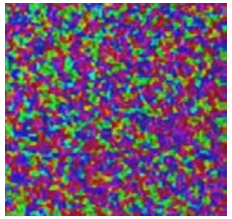
"Concept of manifold and Difference of between AE and VAE"

Concept of manifold



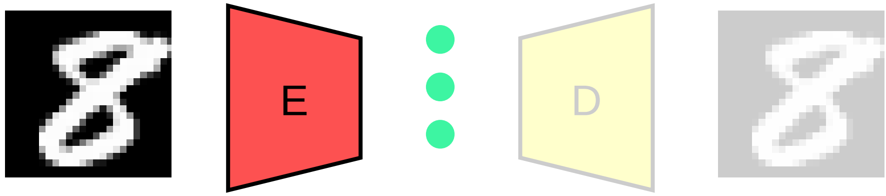
Assumption(manifold hypothesis)

Uniform sampling



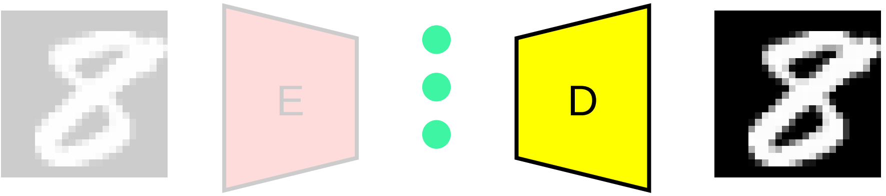
Purpose of AE and VAE

Purpose of AE : Manifold Learning



Unsupervised Learning

Purpose of VAE : Generative Model

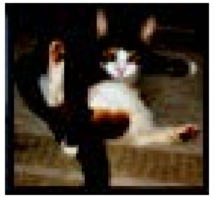


Unsupervised Learning

Appendix

- PyTorch (Variable length inputs)

CIFAR-10



3x32x32

Shape = {Size} torch.Size([128, 3, 32, 32])

Shape = {Size} torch.Size([128, 64, 16, 16])

Shape = {Size} torch.Size([128, 16384])

CelebA

3x178x218

Shape = {Size} torch.Size([128, 3, 218, 178])

Shape = {Size} torch.Size([128, 64, 109, 89])

Shape = {Size} torch.Size([128, 620864])