Machine Learning, Spring 2019

Generative Adversarial Networks (GAN)

Instructor: Prof. Yi Fang
yfang@nyu.edu

Python tutorial: http://learnpython.org/

TensorFlow tutorial: https://www.tensorflow.org/tutorials/

PyTorch tutorial: https://pytorch.org/tutorials/

Acknowledge: The slides are partially referred to the online materials by Taegyun Joen, https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection and online YOLO paper and other materials (from ECS289g by Prof. Lee)

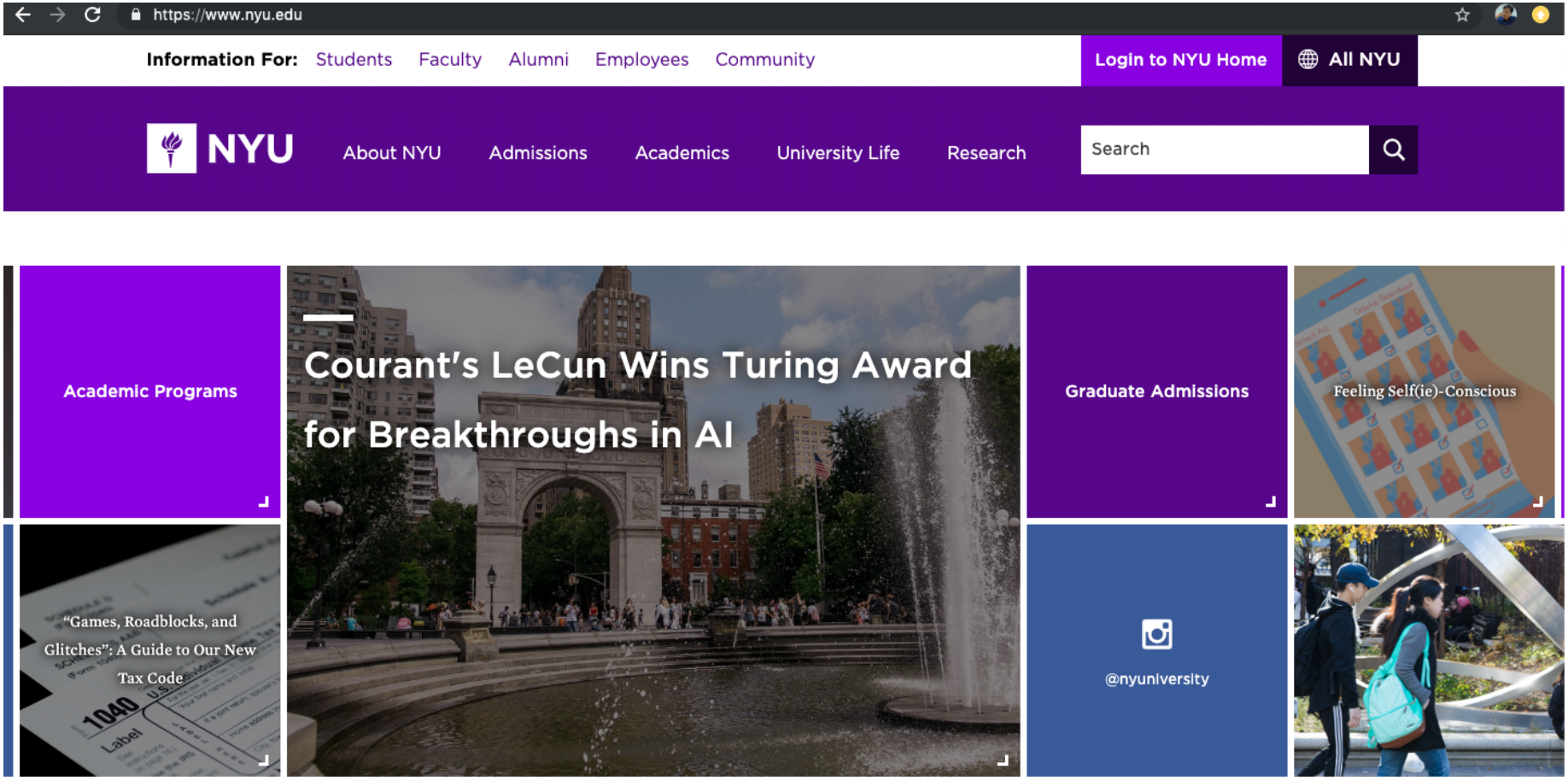
Magic of GANs

- Which one is Computer generated?

Src: Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." arXiv preprint arXiv:1609.04802 (2016).

- ARTISTIC STYLE TRANSFER

Src: https://neurohive.io/en/state-of-the-art/twin-gan-cross-domain-translation-of-human-portraits/


https://www.nyu.edu
Information For: Students Faculty Alumni Employees Community
Login to NYU Home
All NYU
NYU
About NYU
Admissions
Academics
University Life
Research
Search
Academic Programs
Courant's LeCun Wins Turing Award for Breakthroughs in AI
Graduate Admissions
Feeling Self(ie)-Conscious
"Games, Roadblocks, and Glitches": A Guide to Our New Tax Code
@nyuniversity

May 5, 2019

LeCun, NIPS 2016

- Reinforcement learning (cherry)
- Supervised learning (Chocolate)
- Unsupervised/Predictive learning (Cake)
  - Generative adversarial nets (GAN)

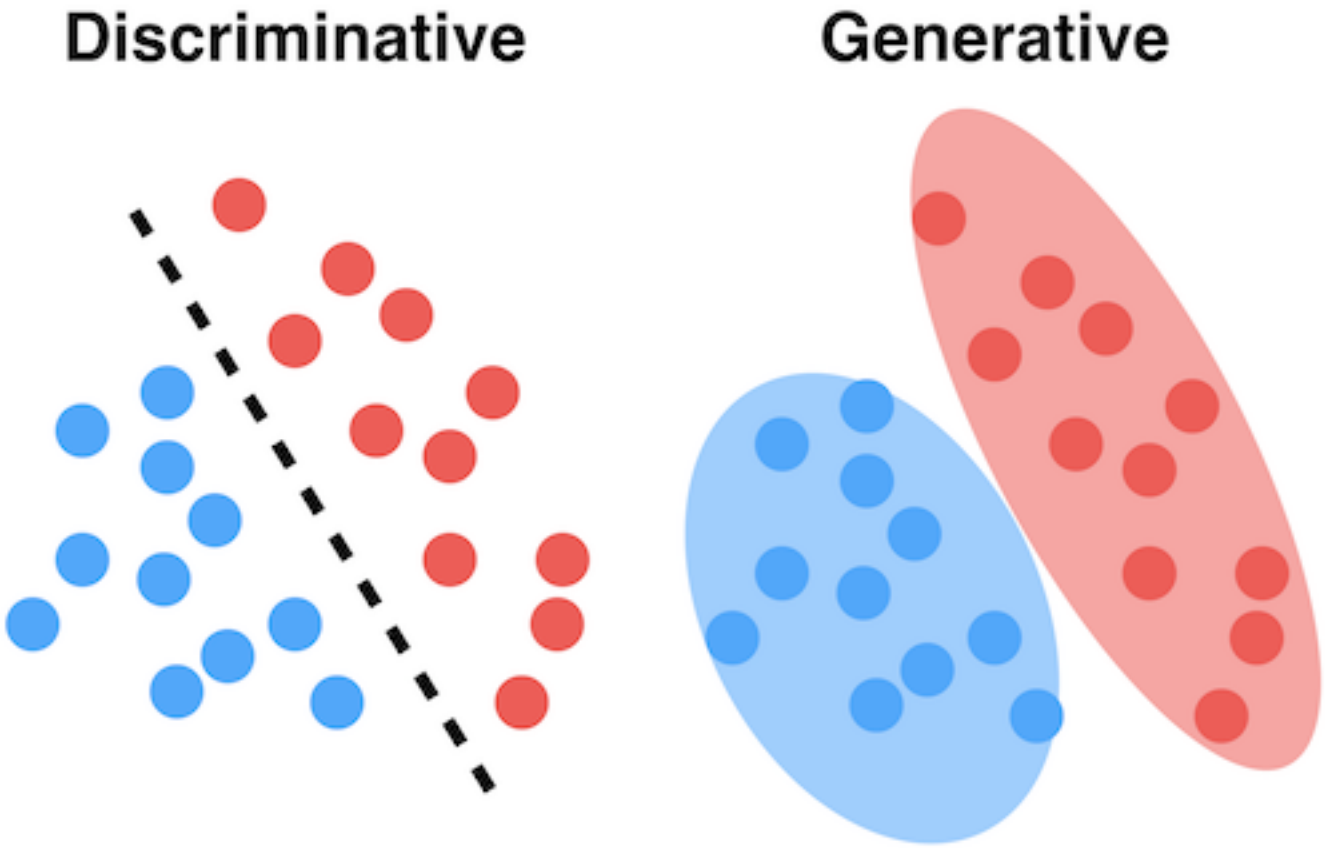
"GANs are the most interesting idea in the last 10 years in ML"

Yann LeCun.

- Introduction of GAN
- GAN applications

- Introduction of GAN
- GAN applications

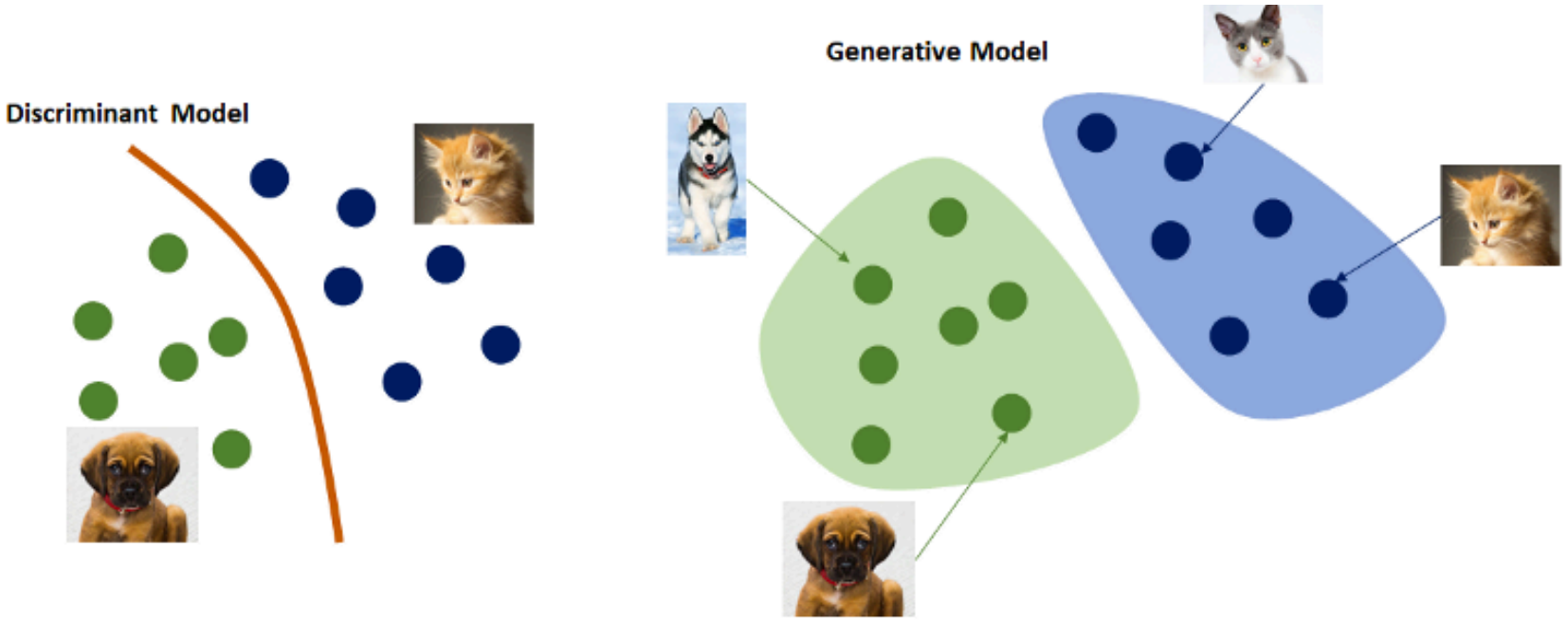
Generative and Discriminative models



Src: https://medium.com/@jordi299/about-generative-and-discriminative-models-d8958b67ad32

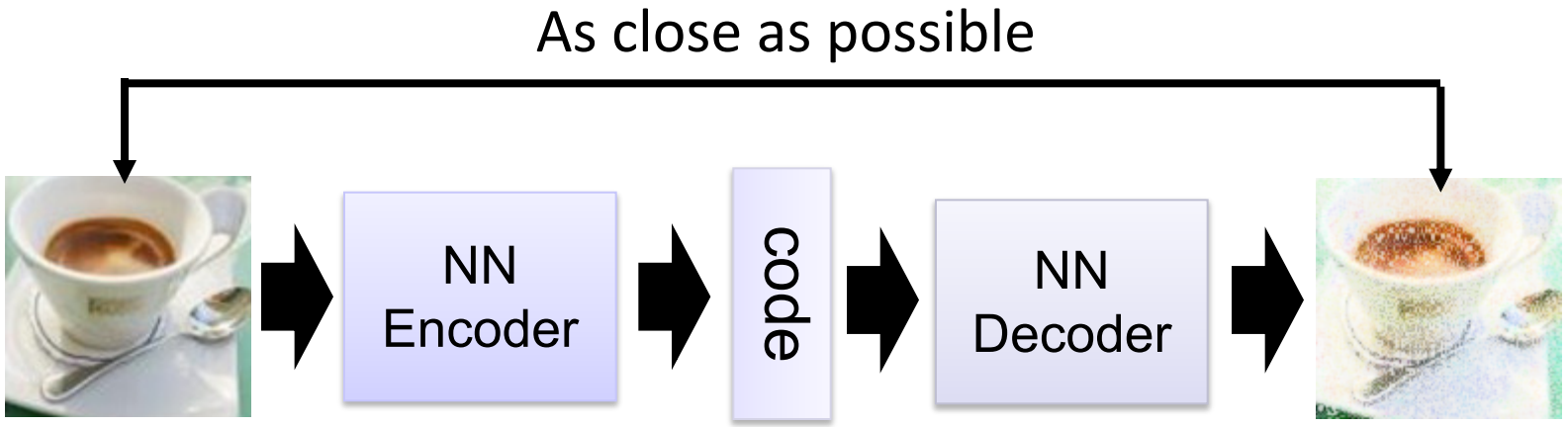
- In supervised learning, we have data x and response (label) y and the goal is to learn a function to map x to y e.g. regression, classification, object detection;

- In unsupervised learning, there are no labels and the goal is to find some underlying hidden structure of the data e.g. clustering, dimensionality reduction, feature learning. The goal of generative models is to generate new samples of data from a distribution. These models are used in problems such as density estimation, a problem of unsupervised learning.

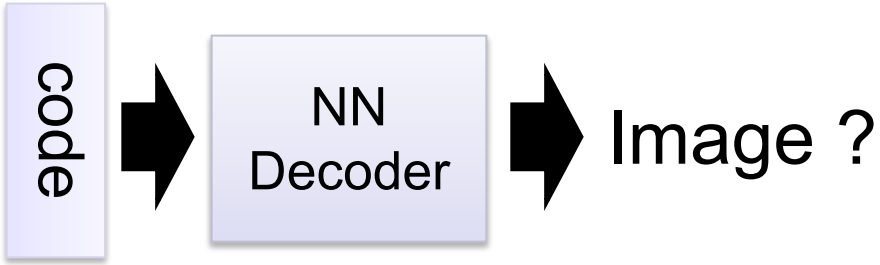


Src: https://kharshit.github.io/blog/2018/09/28/generative-models-and-generative-adversarial-networks

Autoencoder



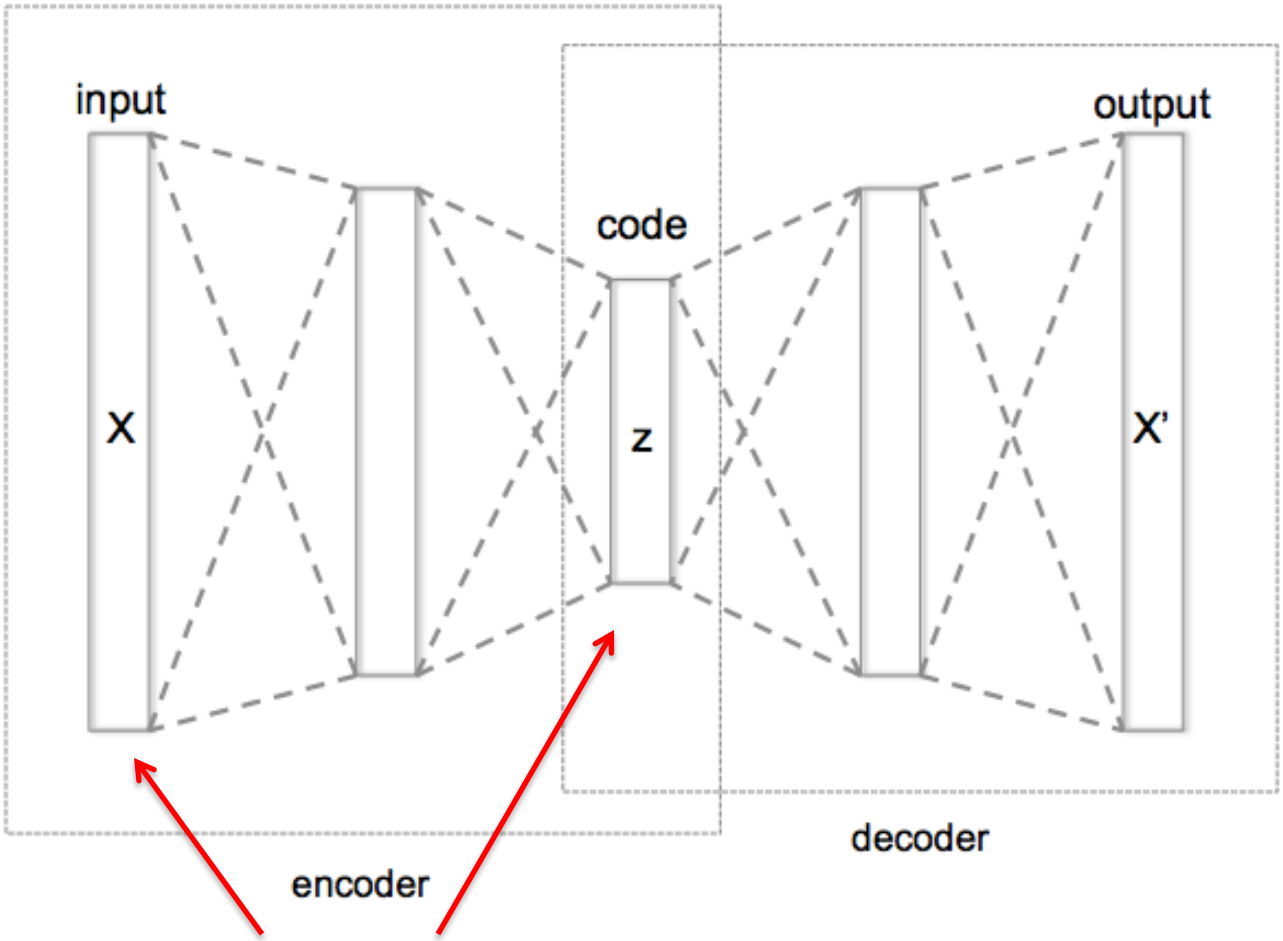
Randomly generate a vector as code



Src: https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture7GAN.ppt

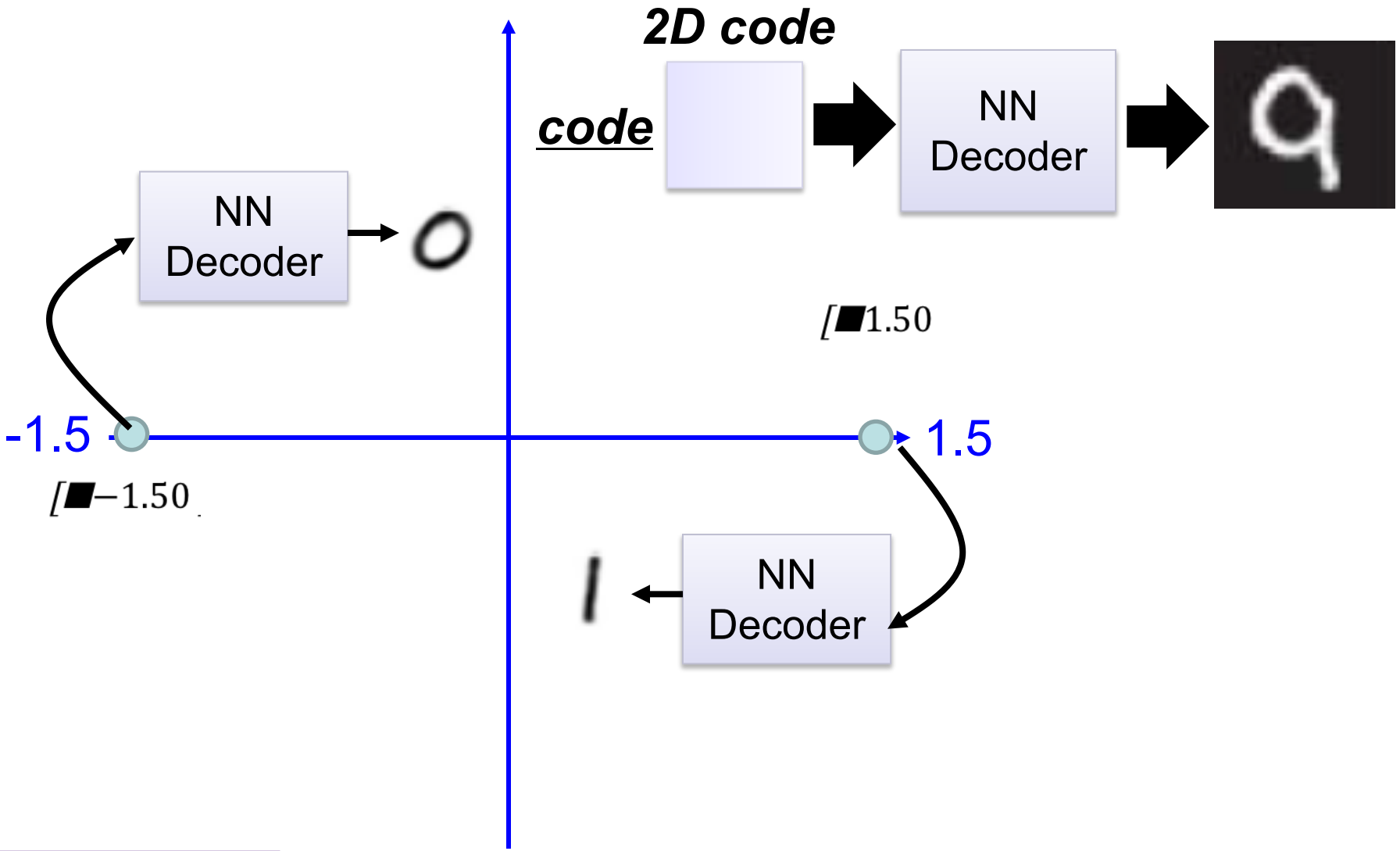
Autoencoder with 3 fully connected layers

Training: model.fit(X,X)
Cost function: $\Sigma_{k=1..N} (x_k - x'_k)^2$

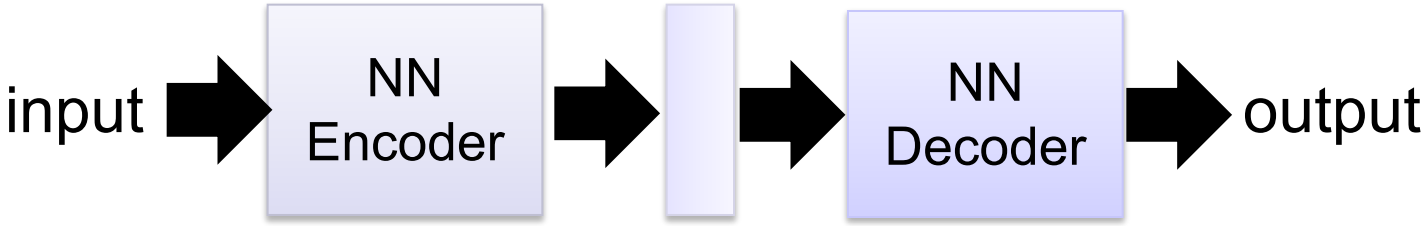


Large → small, learn to compress

Auto-encoder

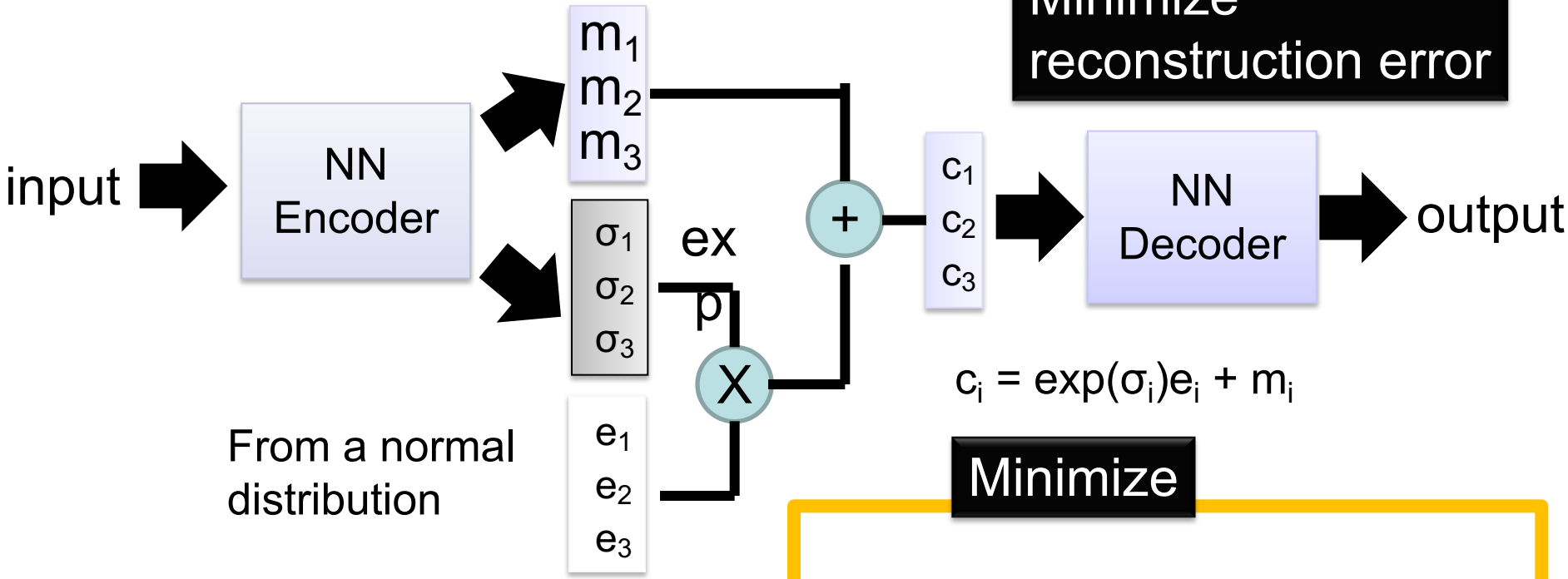


Auto-encoder



code

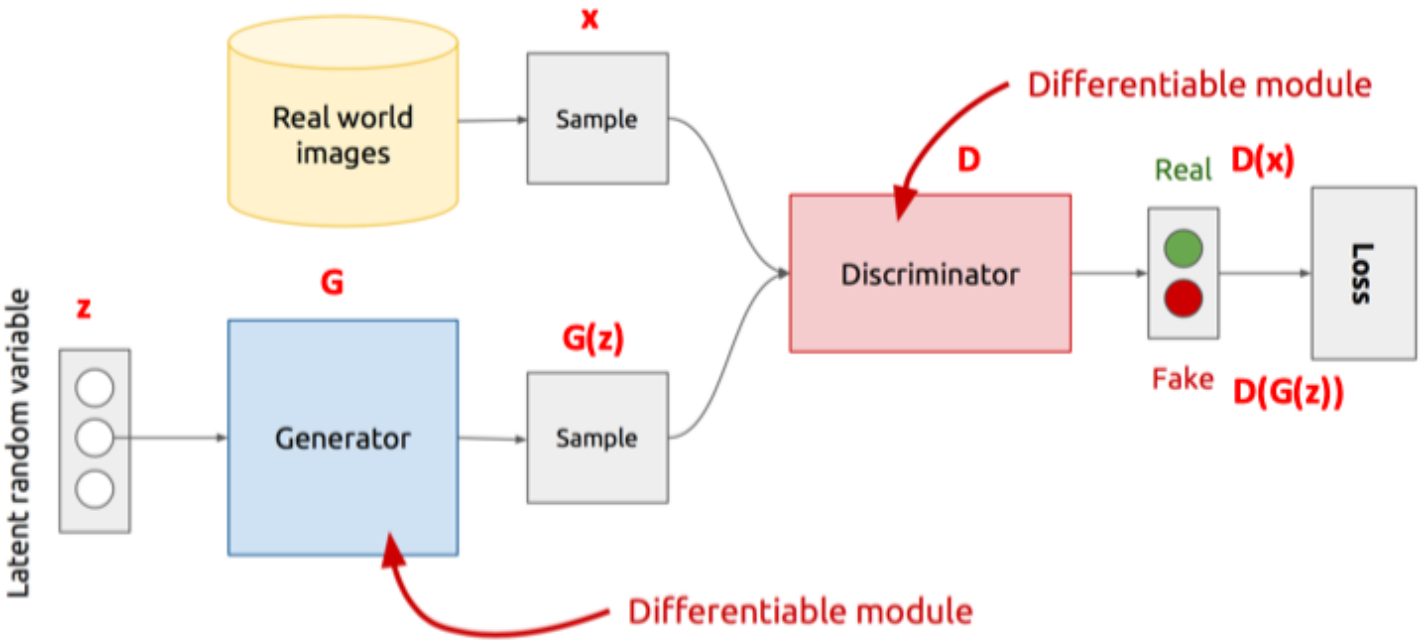
VAE



Auto-Encoding Variational Bayes, https://arxiv.org/abs/1312.6114

Generative Adversarial Network

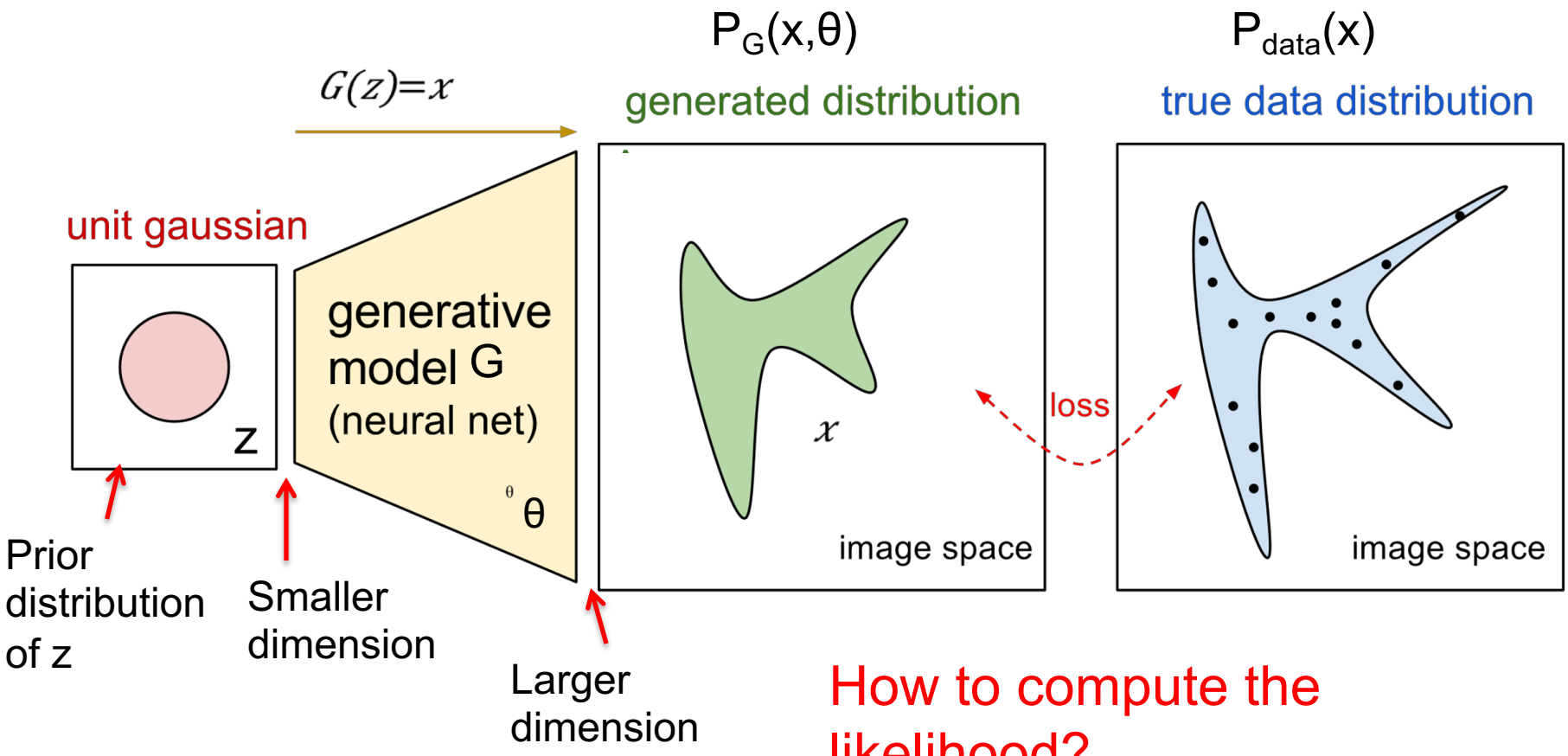
GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Neural Network as $P_G(x; \theta)$

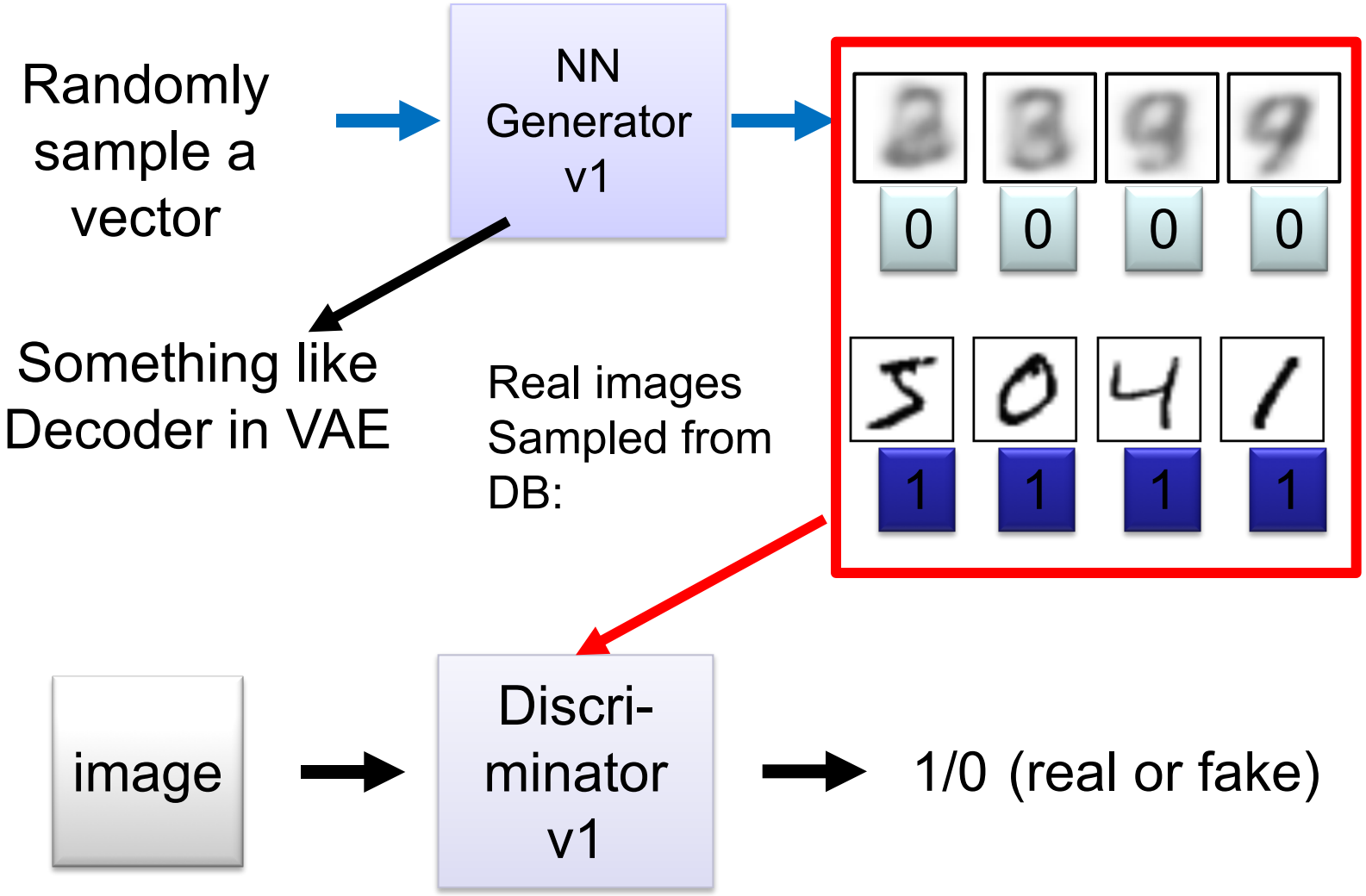


How to compute the likelihood?

$P_G(x) = \text{Integration}_z\ P_{prior}(z)\ I_{[G(z)=x]}dz$

https://blog.openai.com/generative-models/

GAN – Learn a discriminator



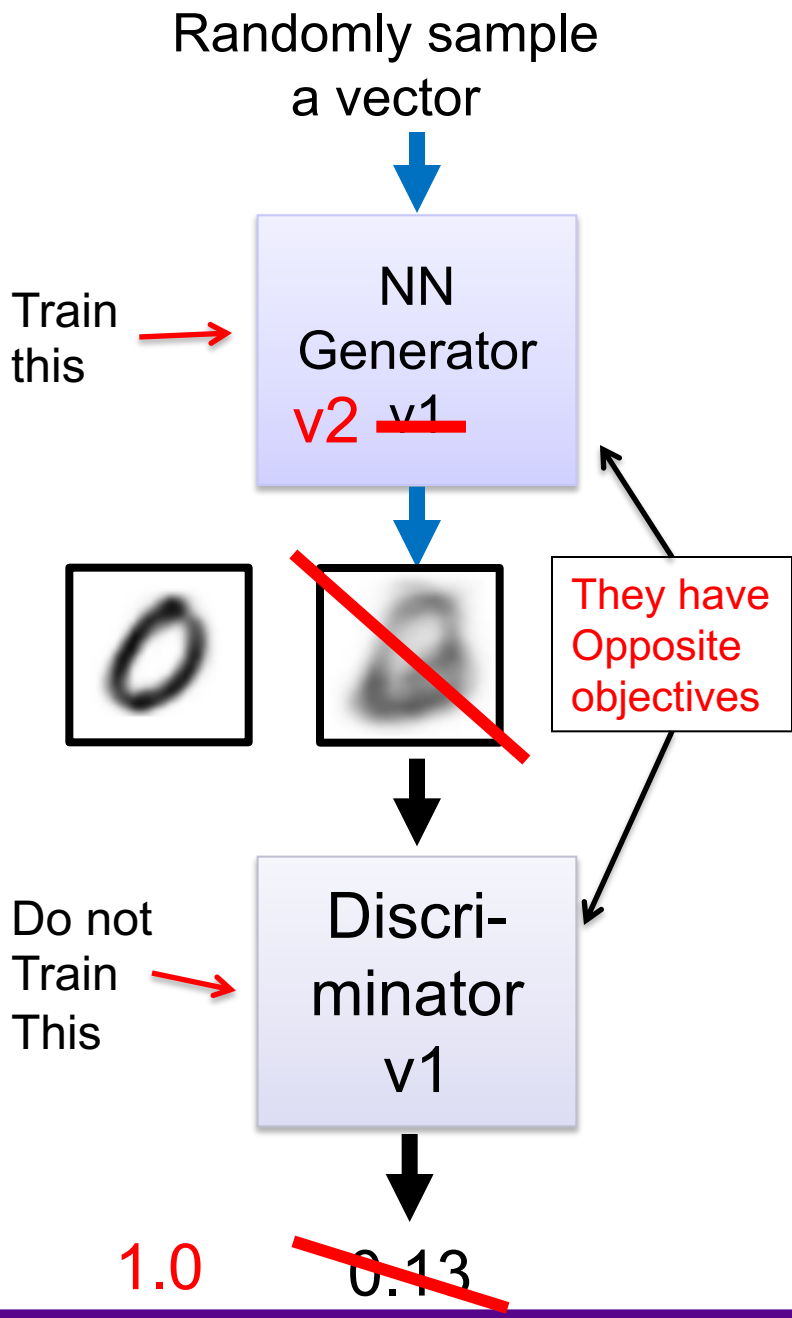
GAN – Learn a generator

Updating the parameters of generator

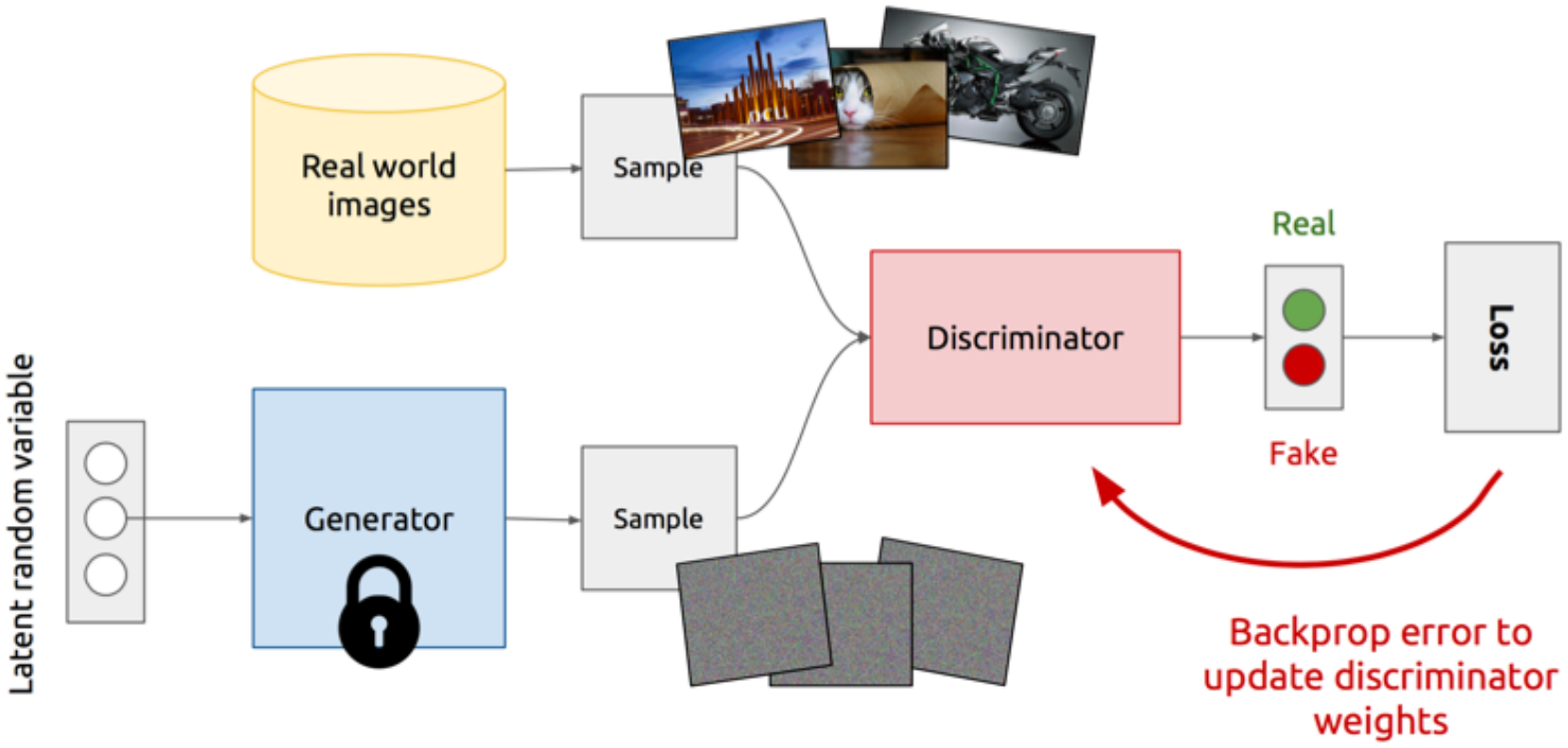
The output be classified as “real” (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator

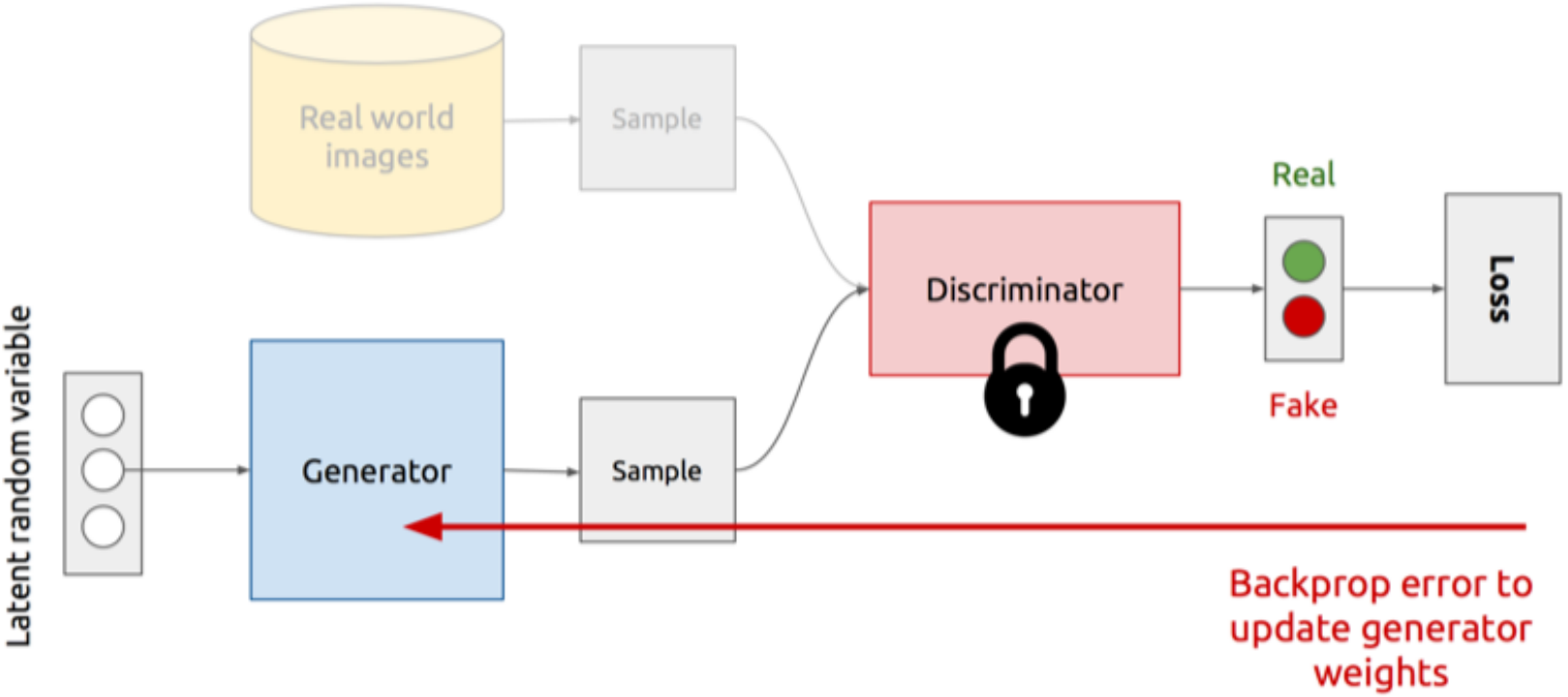


Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN's formulation

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

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \;\; \forall x$

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

Discriminator updates

- 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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

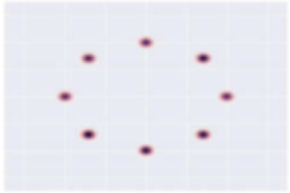
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

GAN Challenges

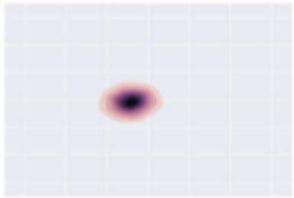
Mode Collapse

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

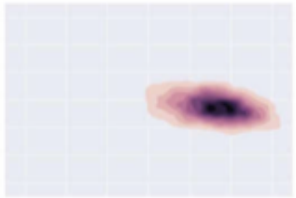
- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point



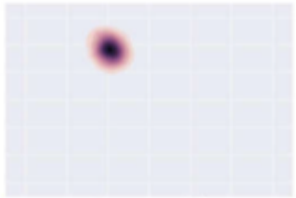
Target



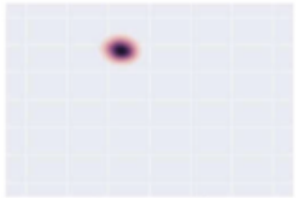
Step 0



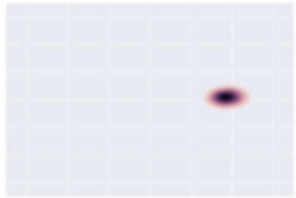
Step 5k



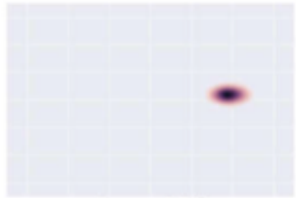
Step 10k



Step 15k



Step 20k



Step 25k

Vanishing gradient strikes back again…

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

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

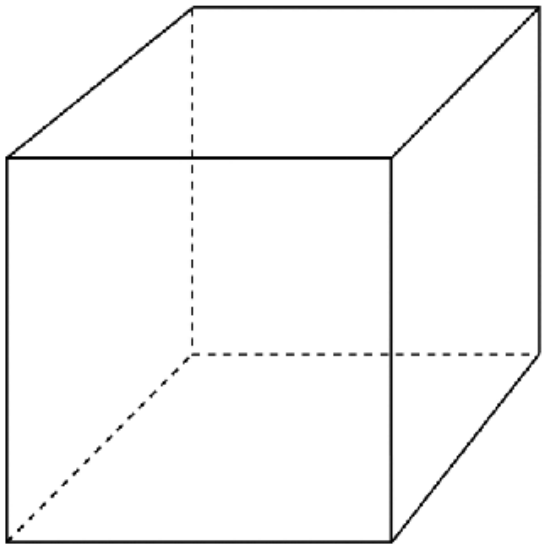
- $\nabla_a \log(1-\sigma(a)) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\,(1-\sigma(a))}{1-\sigma(a)} = -\sigma(a) = -D(G(z))$
- Gradient goes to 0 if D is confident, i.e. $D(G(z)) \to 0$

- Minimize $-\mathbb{E}_{z\sim q(z)}[\log D(G(z))]$ for **Generator** instead (keep Discriminator as it is)

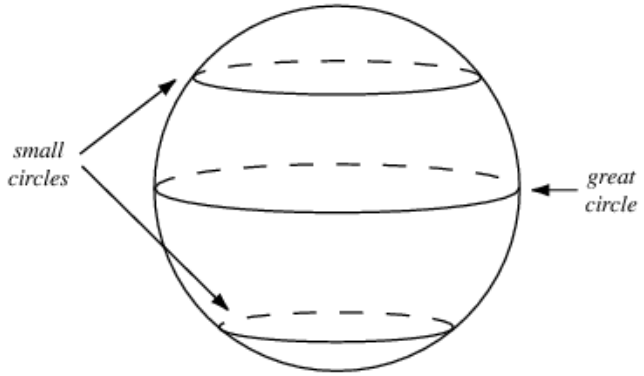
How to Train GAN (Tricks)

- Spherical z for sampling

- Dont sample from a Uniform distribution



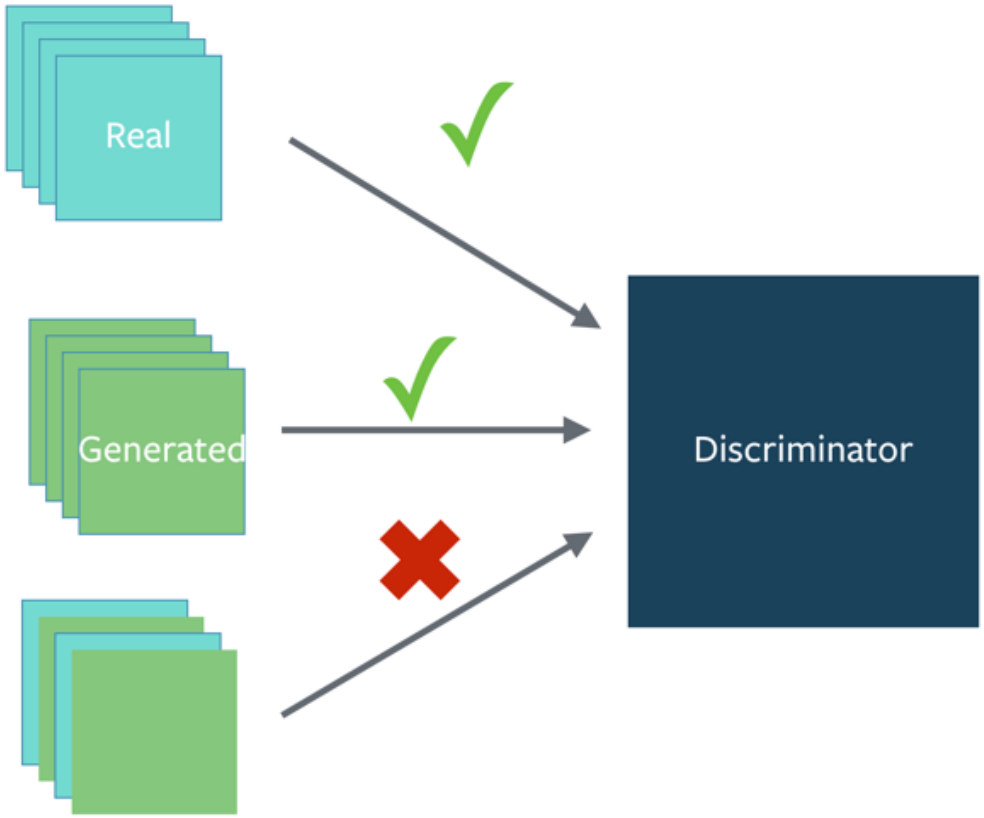
- Sample from a gaussian distribution



Src: https://lanpartis.github.io/deep%20learning/2018/03/12/tricks-of-gans.html

• BatchNorm

- Construct different mini-batches for real and fake, i.e. each mini-batch needs to contain only all real images or all generated images.
- when batchnorm is not an option use instance normalization (for each sample, subtract mean and divide by standard deviation).



- Conditional GANs

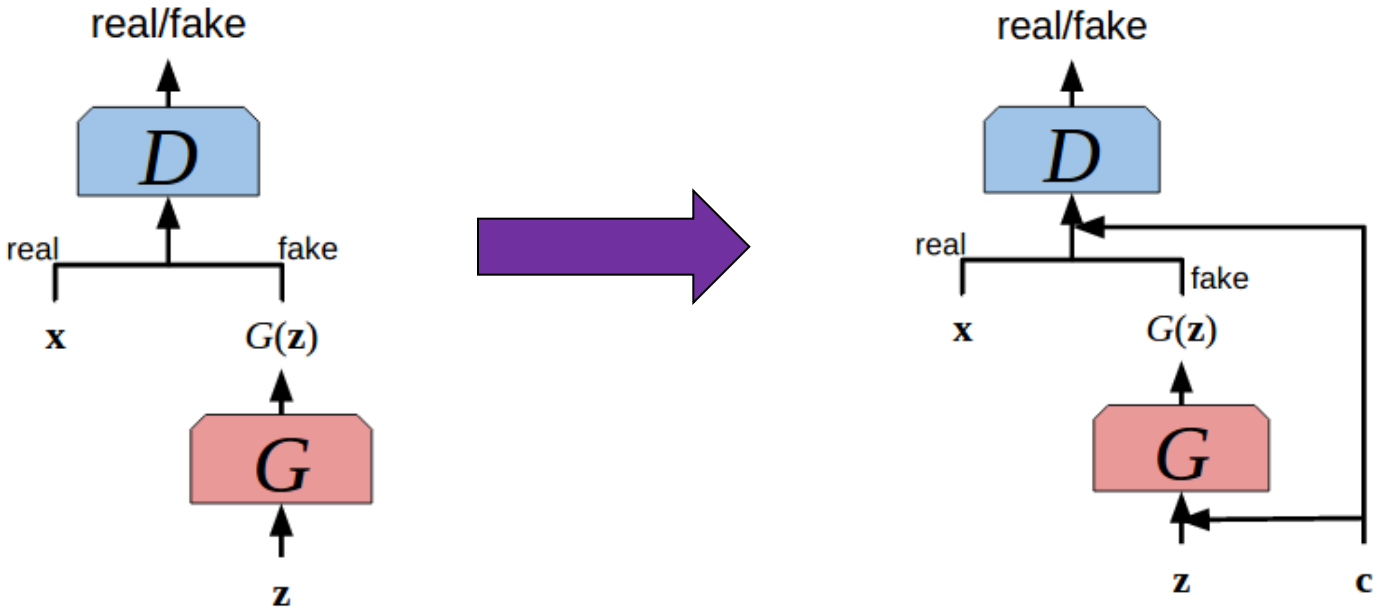


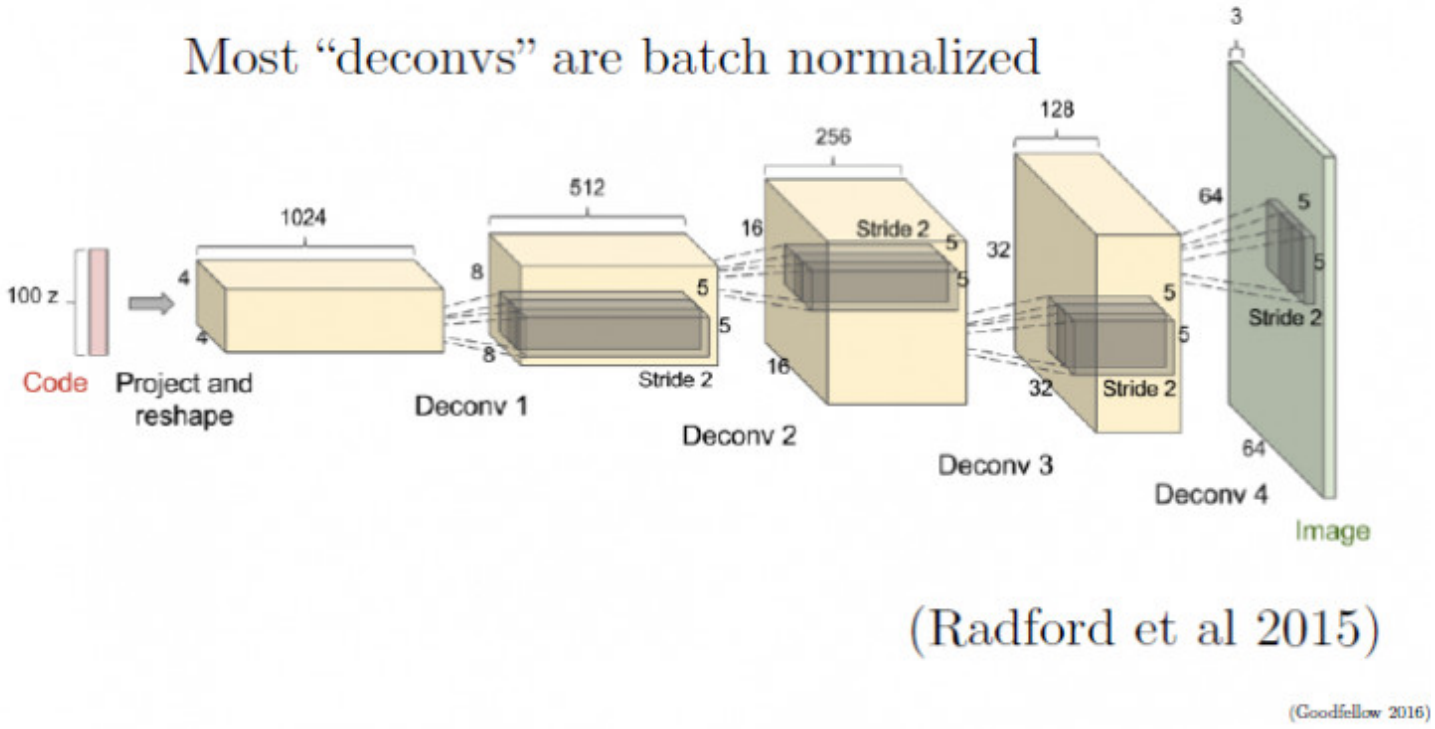
Figure 1: Generative Adversarial Networks

Figure 3: Conditional GAN

Src: https://medium.com/jungle-book/towards-data-set-augmentation-with-gans-9dd64e9628e6

Unsupervised Representation Learning

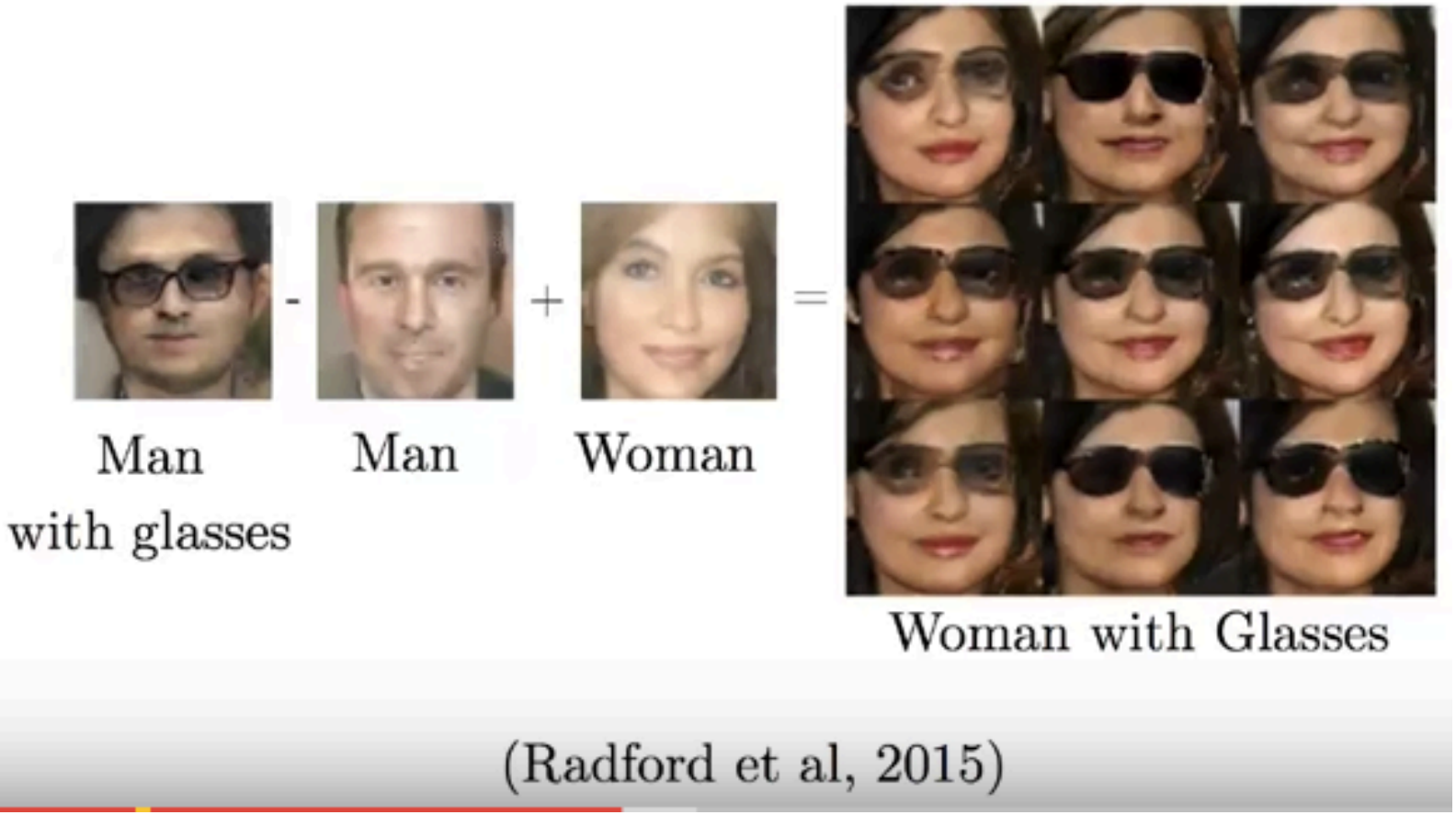
DCGAN Architecture:



DCGAN for LSUN Image Generation:

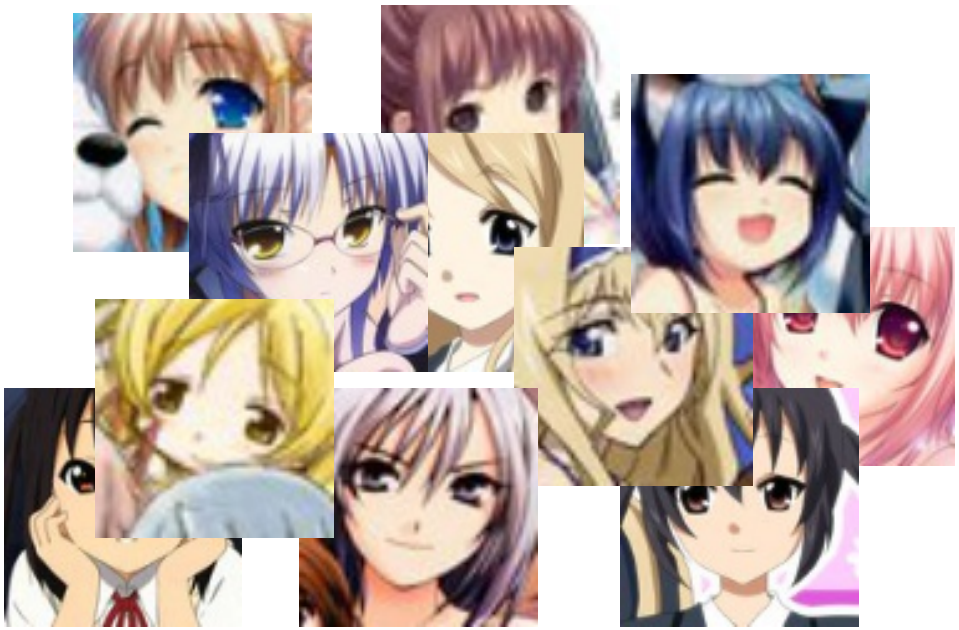
(Radford et al 2015)

DCGAN for Vector Space Arithmetic:



Visualization of Image Generation Process

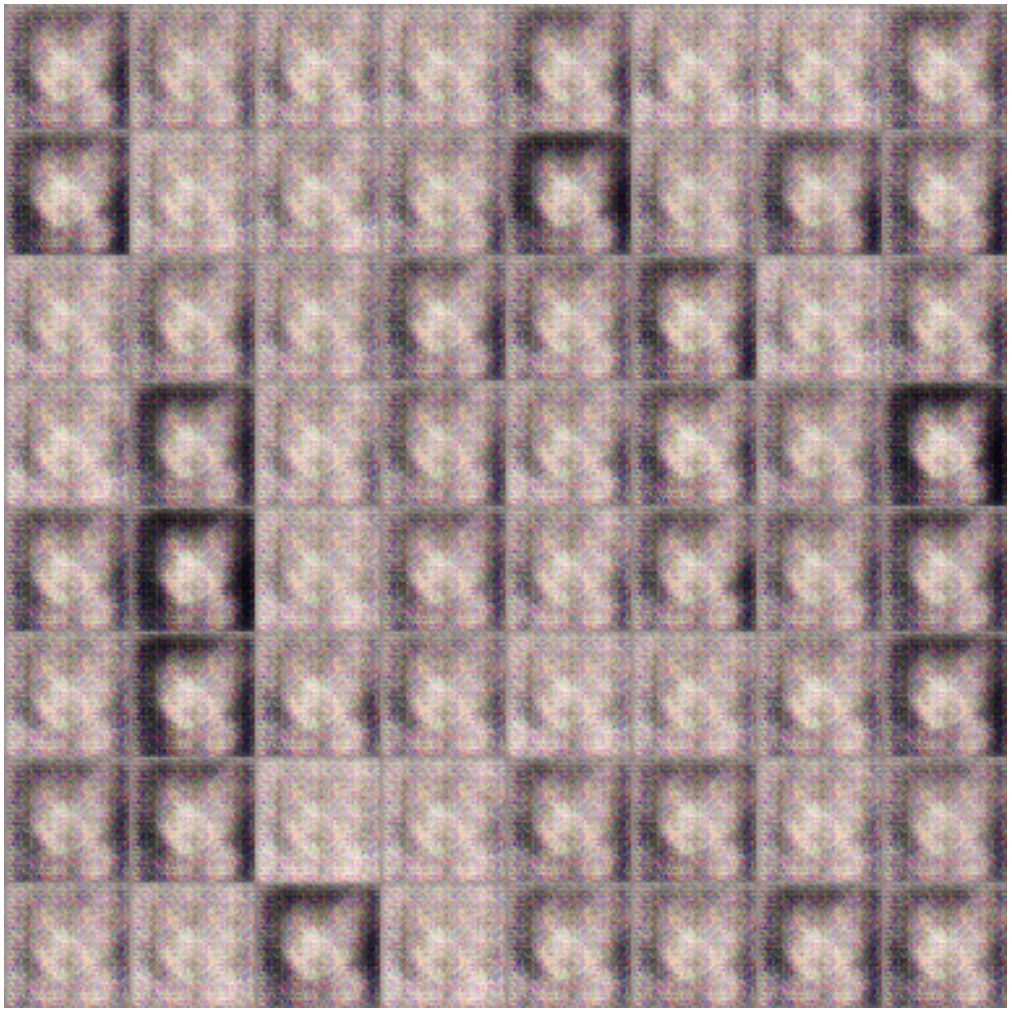
Generating 2nd element figures



Source of images: https://zhuanlan.zhihu.com/p/24767059
From Dr. HY Lee's notes.

DCGAN: https://github.com/carpedm20/DCGAN-tensorflow

GAN – generating 2nd element figures



100 rounds

This is fast, I think you can use your CPU

GAN – generating 2nd element figures

1000 rounds

GAN – generating 2nd element figures

2000 rounds

GAN – generating 2nd element figures

5000 rounds

GAN – generating 2nd element figures

10,000 rounds

GAN – generating 2nd element figures

20,000 rounds

GAN – generating 2nd element figures

50,000 rounds

- Introduction of GAN

- GAN applications

Applications of GAN

- Adversarial Learning
- Low Level Vision
- Domain Adaptation
- Image translation

References

- http://slazebni.cs.illinois.edu/spring17/
- https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture7GAN.ppt