ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Generative Models

Cihan Öngün
Prof. Dr. Alptekin Temizel

Graduate School of Informatics
METU

- Unsupervised Learning

- Generative Models
  - Autoregressive Models (PixelRNN, PixelCNN)
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Learning

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Learning

$$\min_{\theta \in \mathcal{M}} d(p_{\text{data}}, p_\theta)$$

What is the representation for the model family $\mathcal{M}$?

What is the objective function $d(\cdot)$?

What is the optimization procedure for minimizing $d(\cdot)$?

Stanford's CS236 https://deepgenerativemodels.github.io/notes/introduction/]

Inference

1. *Density estimation:* Given a datapoint $\mathbf{x}$, what is the probability assigned by the model, i.e., $p_\theta(\mathbf{x})$?

2. *Sampling:* How can we *generate* novel data from the model distribution, i.e., $\mathbf{x}_{\text{new}} \sim p_\theta(\mathbf{x})$?

3. *Unsupervised representation learning:* How can we learn meaningful feature representations for a datapoint $\mathbf{x}$?

Stanford's CS236 https://deepgenerativemodels.github.io/notes/introduction/]

Generating New Data

Novel: The generated data must be different from any samples of input data.

- Measure the similarity of every generated data with every sample in input data.
- If it is not different, system just copies the input data

Same distribution: The generated data must be in the same distribution with the input data.

- Same structure, class and concept of input data.
- Generated data would be out of scope and useless

Meaningful: Difficult to measure.

- Visual perception
- Meaningful images or objects for human perception.
- Even all of the metrics and mathematical outcomes seem well; the generated data is not successful if it is not meaningful for human perception.

Taxonomy of Generative Models

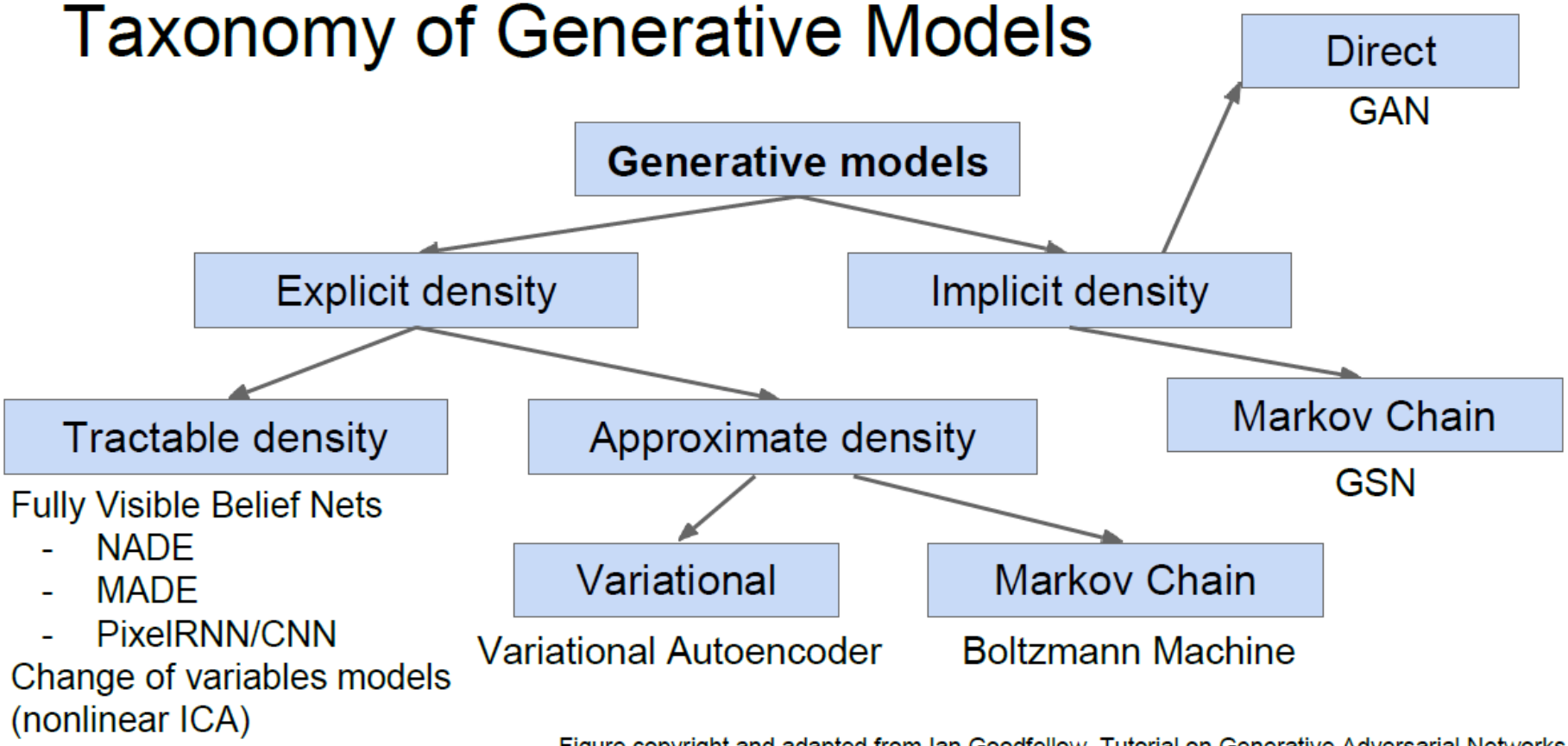
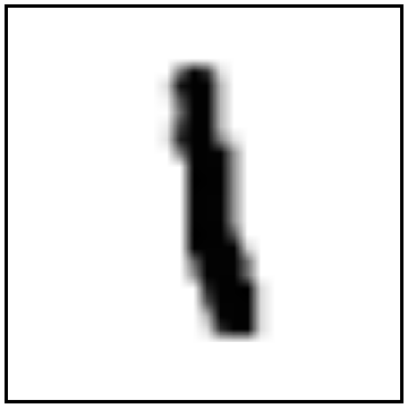
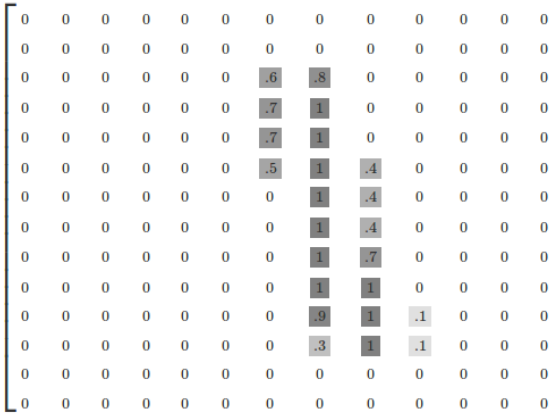


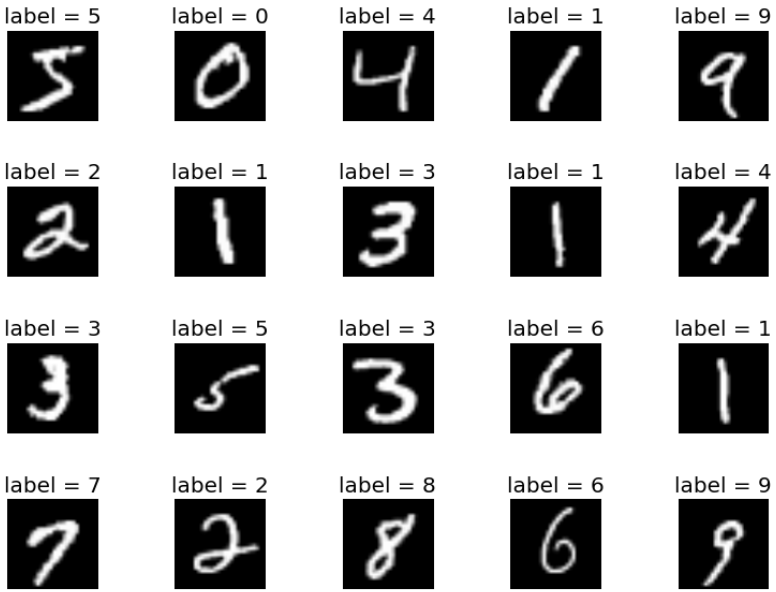
Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Autoregressive Models



≃





$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Using Chain Rule

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$$

Fully General, no assumptions needed (exponential size, no free lunch)

Bayes Net

$$p(x_1, x_2, x_3, x_4) \approx p_{\text{CPT}}(x_1)p_{\text{CPT}}(x_2 \mid x_1)p_{\text{CPT}}(x_3 \mid \cancel{x_1}, x_2)p_{\text{CPT}}(x_4 \mid x_1, \cancel{x_2, x_3})$$

Assumes conditional independencies; tabular representations via conditional probability tables (CPT)

Neural Models

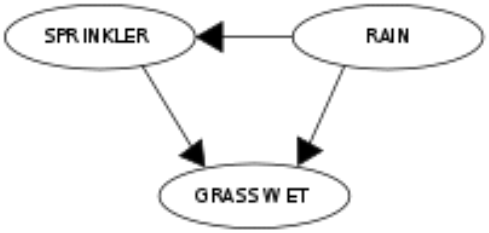
$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2 \mid x_1)p_{\text{Neural}}(x_3 \mid x_1, x_2)p_{\text{Neural}}(x_4 \mid x_1, x_2, x_3)$$

Assumes specific functional form for the conditionals. A sufficiently deep neural net can approximate any function.

Stanford's CS236 https://deepgenerativemodels.github.io/notes/introduction/]

x1	x2	x3
x4	x5	x6
x7	x8	x9

SPRINKLER		
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



RAIN	
T	F
0.2	0.8

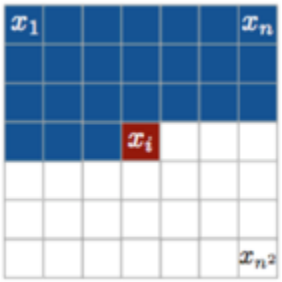
		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$$

$$p(x_1, x_2, x_3, x_4) \approx p_{\mathrm{CPT}}(x_1)p_{\mathrm{CPT}}(x_2 \mid x_1)p_{\mathrm{CPT}}(x_3 \mid \cancel{x_1}, x_2)p_{\mathrm{CPT}}(x_4 \mid x_1, \cancel{x_2, x_3})$$

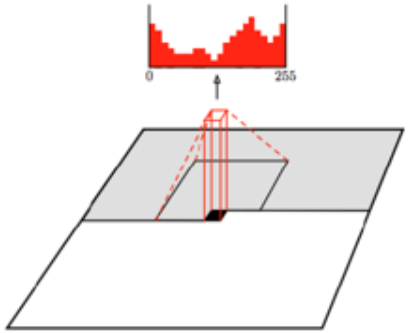
$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2 \mid x_1)p_{\mathrm{Neural}}(x_3 \mid x_1, x_2)p_{\mathrm{Neural}}(x_4 \mid x_1, x_2, x_3)$$

PixelRNN (Oord et al., 2016)

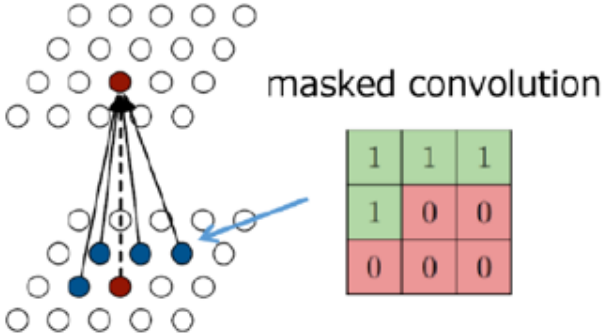


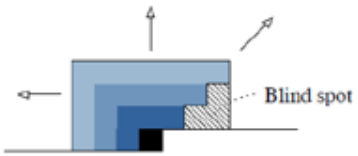
Stanford's CS236 https://deepgenerativemodels.github.io/notes/introduction/]

PixelCNN (Oord et al., 2016)



Idea: Use convolutional architecture to predict next pixel given context (a neighborhood of pixels).
Challenge: Has to be autoregressive. Masked convolutions preserve raster scan order. Additional masking for colors order.





Stanford's CS236 https://deepgenerativemodels.github.io/notes/introduction/]

PixelCNN (Oord et al., 2016)

PolyGen

An Autoregressive Generative Model of 3D Meshes

Class conditional samples

Autoregressive Models Summary

+ Can explicitly compute likelihood p(x)

+ Likelihood is good evaluation metric

+ Easy to extend to continuous domain

- Need to find a sequential ordering if data is not sequential

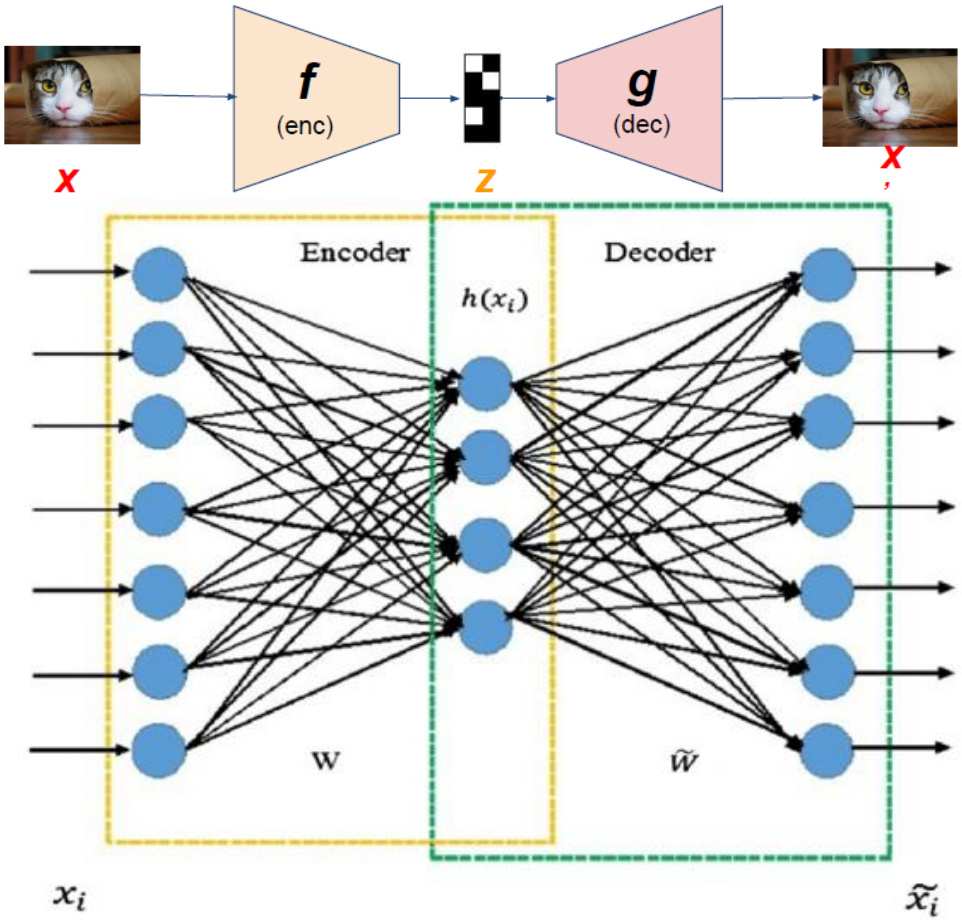
- Sequential generation is very slow

- Low resolution generated samples, poor edges

- No natural way to get features, cluster points, do unsupervised learning

Autoencoders (Reminder)

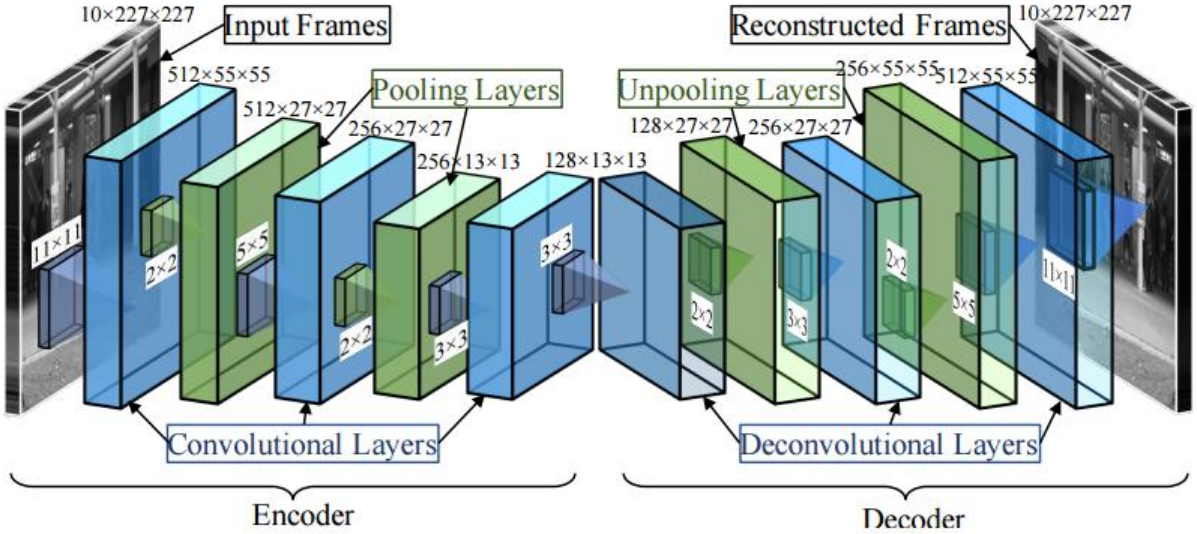
- Neural networks, intended to reproduce output
- Encoder/Decoder architecture



Autoencoders (Reminder)

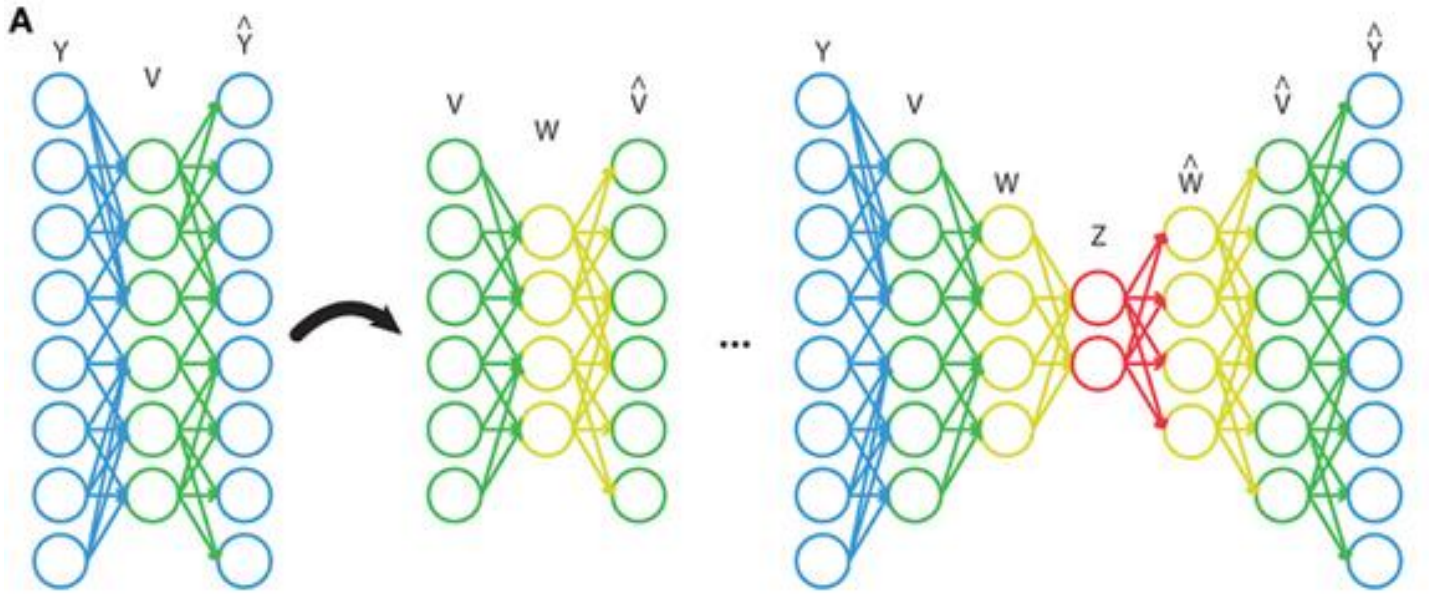
Input image	2-D latent space	5-D latent space	10-D latent space
			

DC Autoencoders (Reminder)



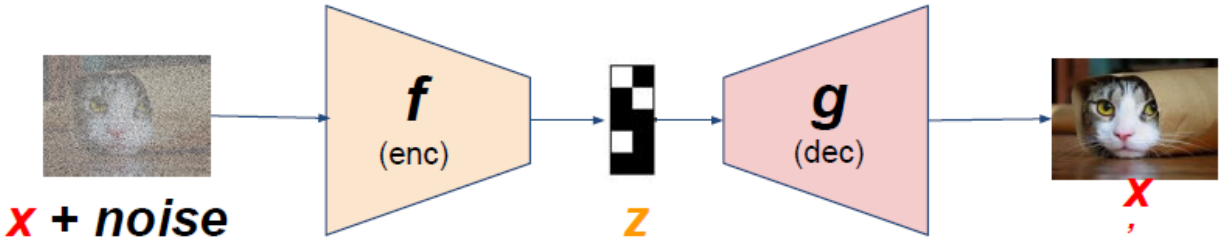
Stacked Autoencoders (Reminder)

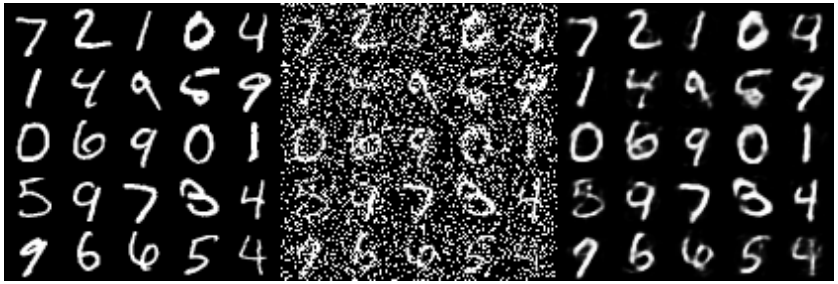
- Multilayer Autoencoders
- Mostly trained layer by layer



Denoising Autoencoders (Reminder)

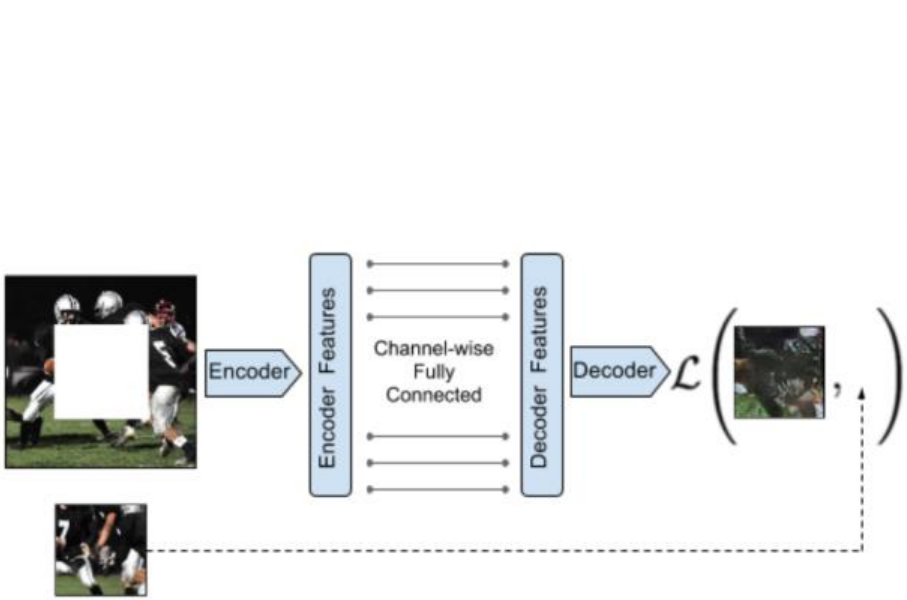
- Add noise to input
- Calculate output loss with original image

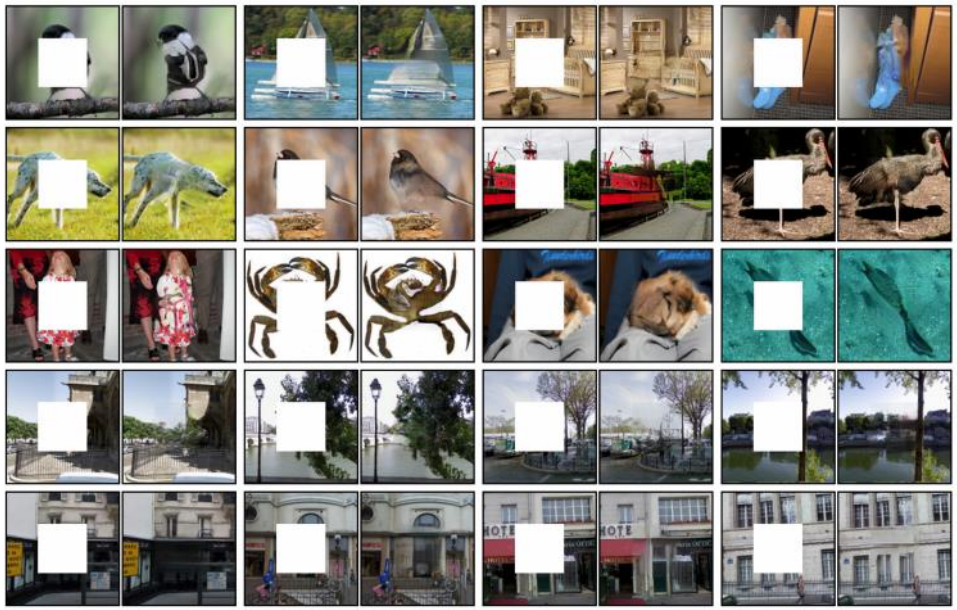




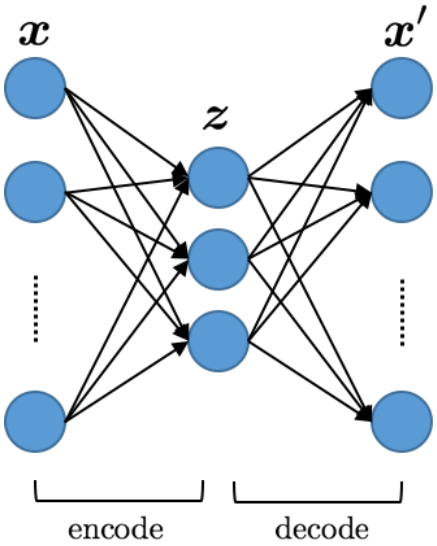
Context Autoencoders (Reminder)

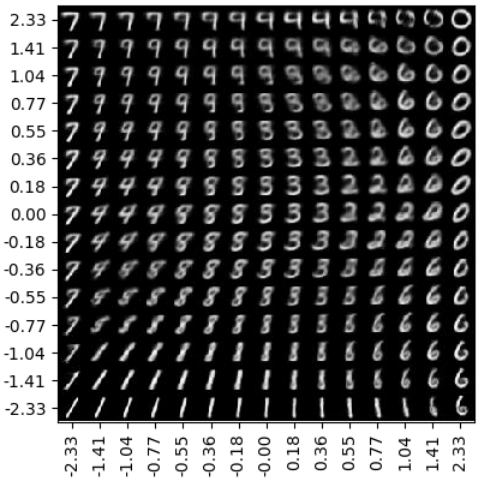
- Crop a part of the image
- Calculate output loss with cropped part

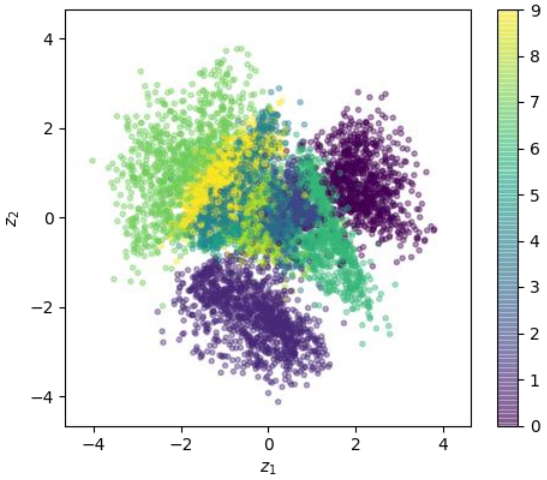




Latent space

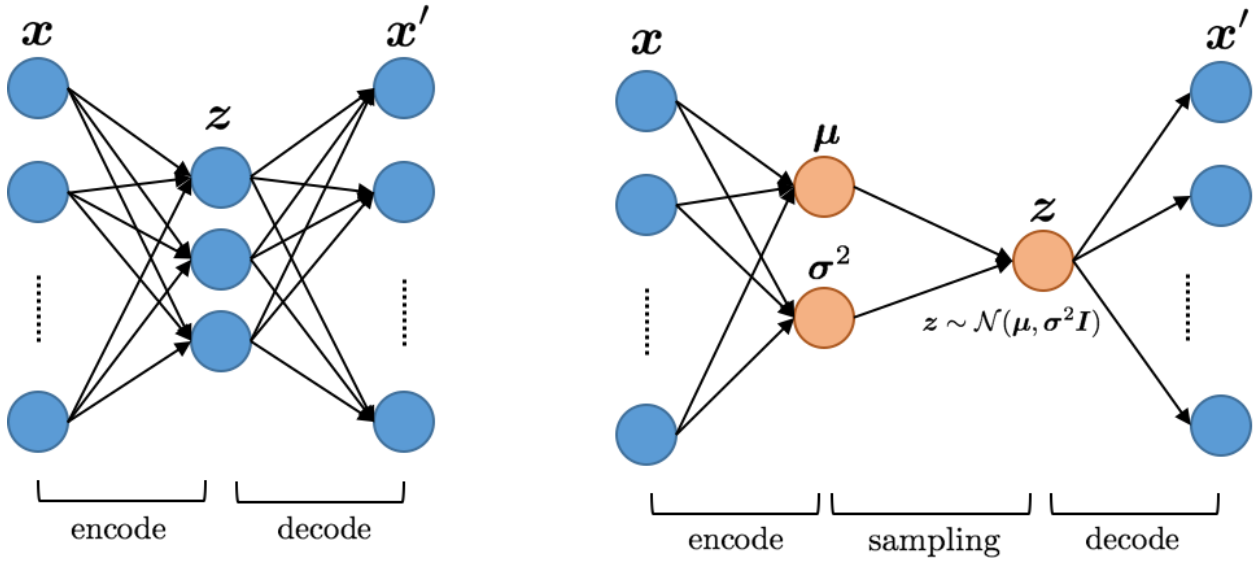




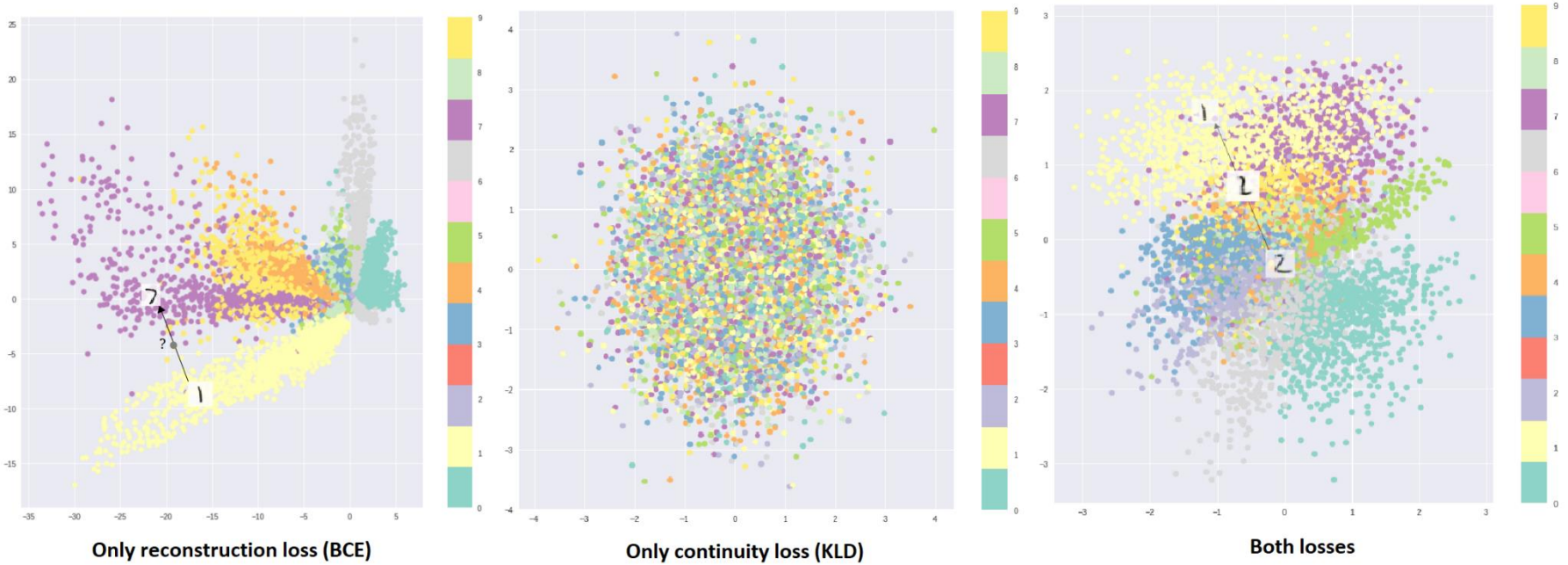


Variational Autoencoders

- Generating new images
- Prevent overfitting
- Extract meaningful latent variables



Variational Autoencoders



Only reconstruction loss (BCE)

Only continuity loss (KLD)

Both losses

Variational Autoencoders

How should we measure distance between distributions?

The **Kullback-Leibler divergence** (KL-divergence) between two distributions p and q is defined as

$$D(p\|q) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}.$$

Notice that KL-divergence is **asymmetric**, i.e., $D(p\|q) \neq D(q\|p)$

$$\begin{aligned} \mathbf{D}(P_{\text{data}}||P_\theta) &= \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log\left(\frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}\right)\right] \\ &= \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_{\text{data}}(\mathbf{x})\right] - \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right] \end{aligned}$$

Variational Autoencoders

$$\mathcal{L}^{(m)} = \frac{1}{M}\sum_{i=1}^{M}(\mathcal{L}_{latent}^{(i)} + \mathcal{L}_{recon}^{(i)})$$

$$\mathcal{L}_{recon}^{(i)} = -\sum_{k=1}^{K}(y_k^{(i)} - x_k^{(i)})^2$$

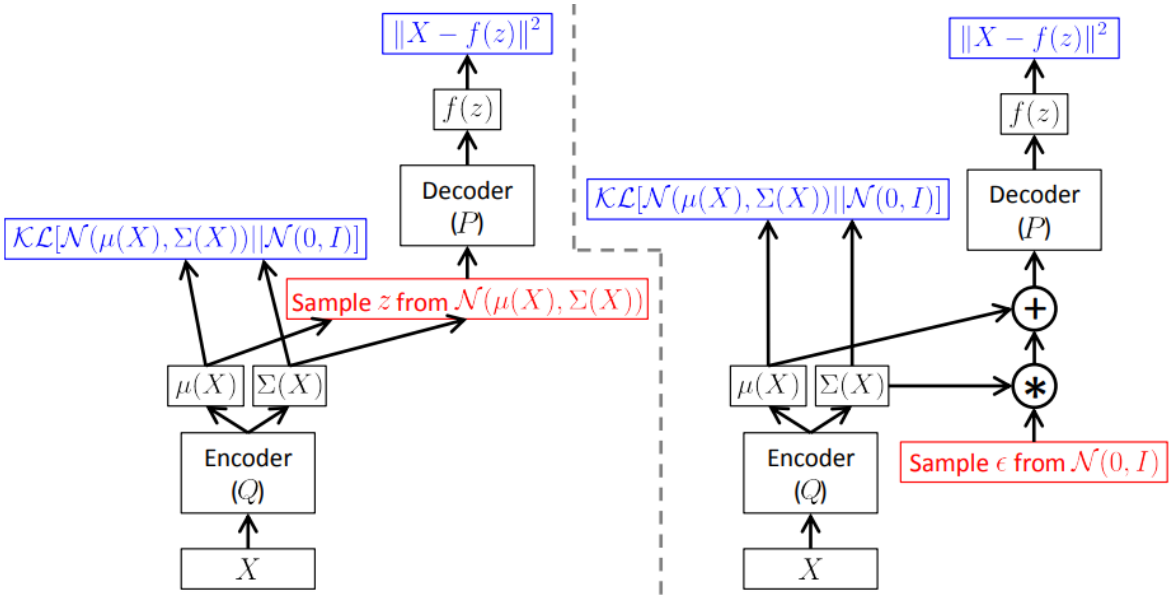
$$D_{KL}[N(\mu(X), \Sigma(X)) \| N(0,1)] = \frac{1}{2}\sum_{k}\left(\exp(\Sigma(X)) + \mu^2(X) - 1 - \Sigma(X)\right)$$

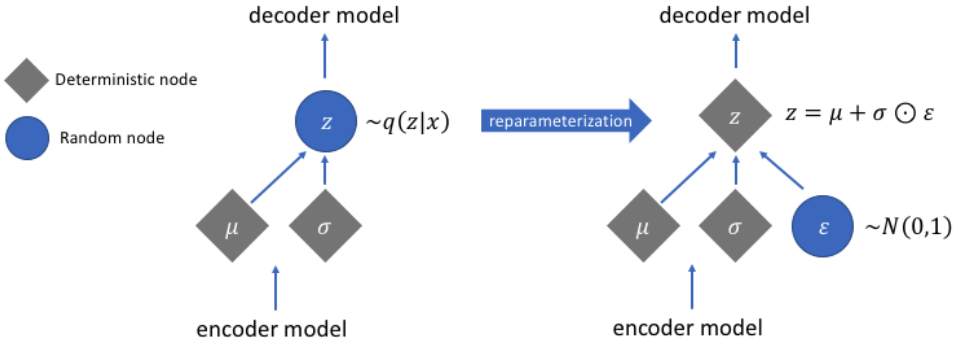
$$\mathcal{L}_{latent}^{(i)} = -\frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j^{(i)})^2 - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2)$$

```
kl_loss = -0.5 * tf.reduce_sum(1 + log_sigma_sq - tf.square(mu) - tf.exp(log_sigma_sq), axis=1)
```

Deriving the Standard Variational Autoencoder (VAE) Loss Function https://arxiv.org/pdf/1907.08956.pdf

Variational Autoencoder Training





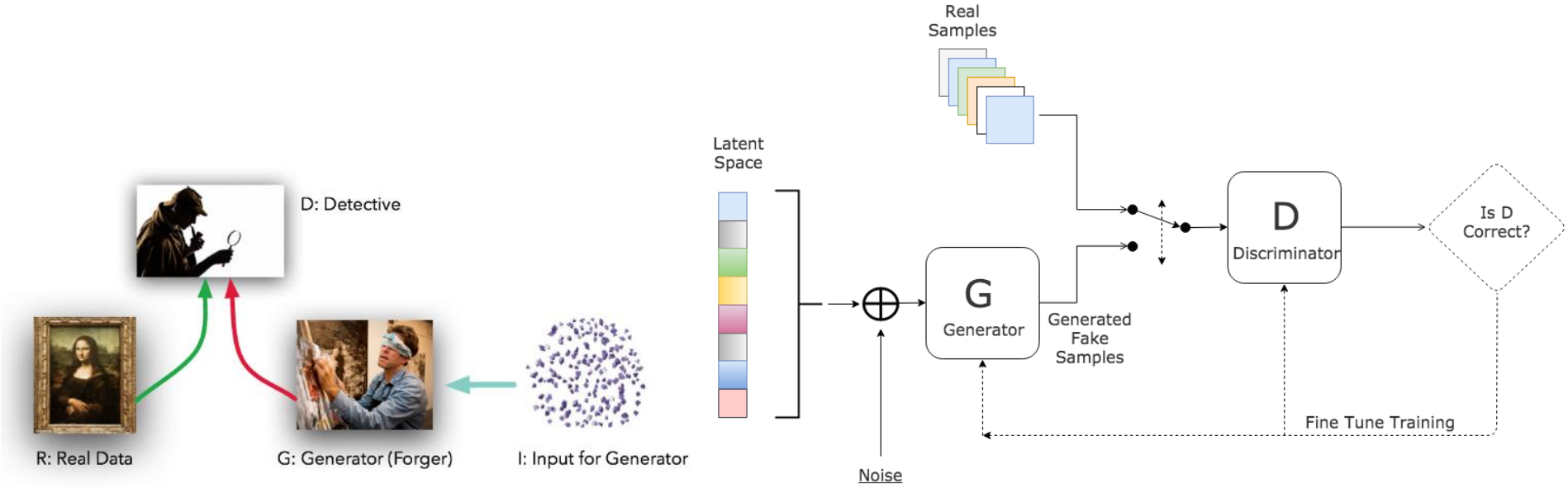
Tutorial on Variational Autoencoders https://arxiv.org/pdf/1606.05908.pdf

β-VAE

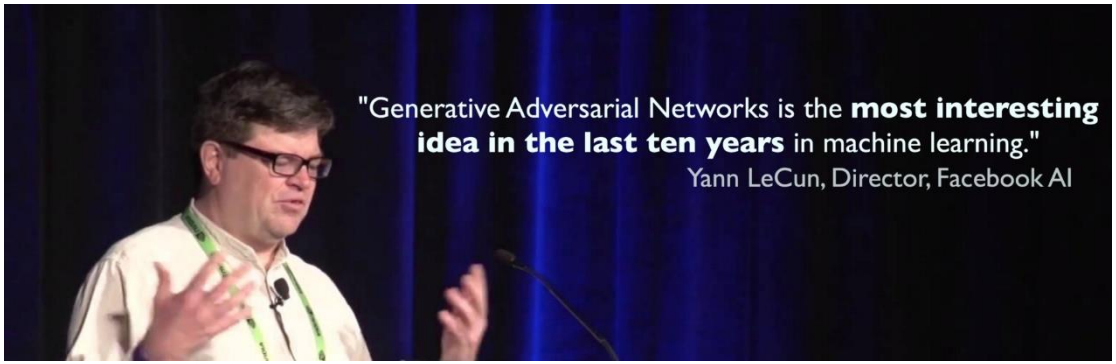
$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta \, D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

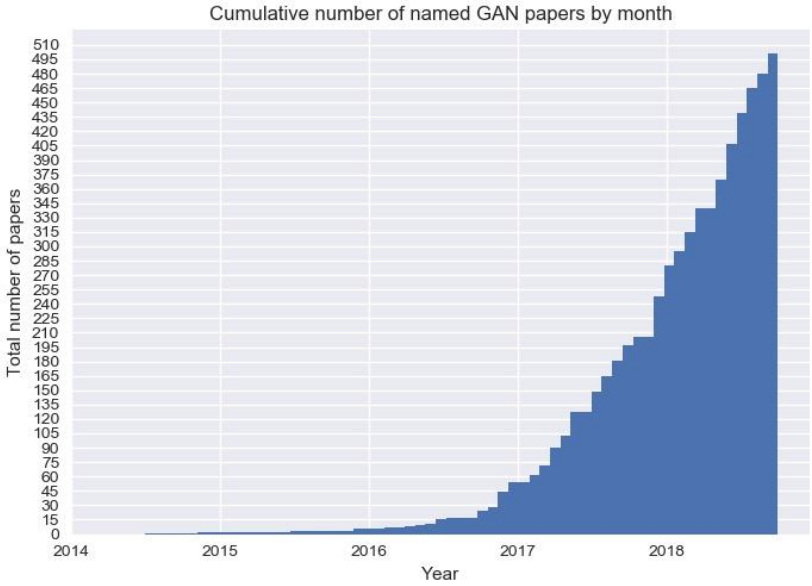
Generative Adversarial Networks (GAN)

- "Generative Adversarial Nets",10 June 2014, Ian Goodfellow (Google Brain Research Team, OpenAI, Apple)
- An adversarial process for estimating generative models
- Consists of 2 simultaneously trained models : a generative model G and a discriminator model D.
- The generative model G takes random noise as input and generates data candidates.
- Discriminator model D tries to distinguish which is real data.



Generative Adversarial Networks

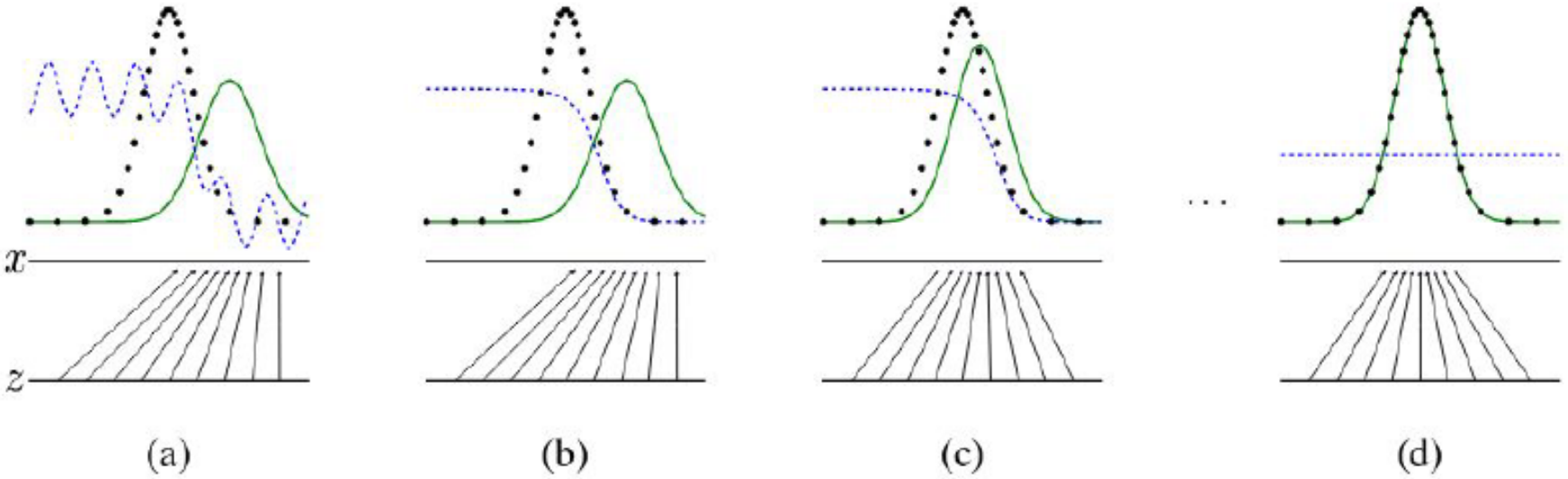




GAN example

- 2D data, MNIST dataset, handwritten digits

GAN Training



Training stages of a GAN
Black dotted: True data
Green solid: Generated data
Blue dotted: Discriminator loss

GAN Training – Loss Function

- Loss function proposed in Goodfellow's paper introducing GANs

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

True data

Noise provided for generating data

GAN training - Workflow

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**

  - 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:

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

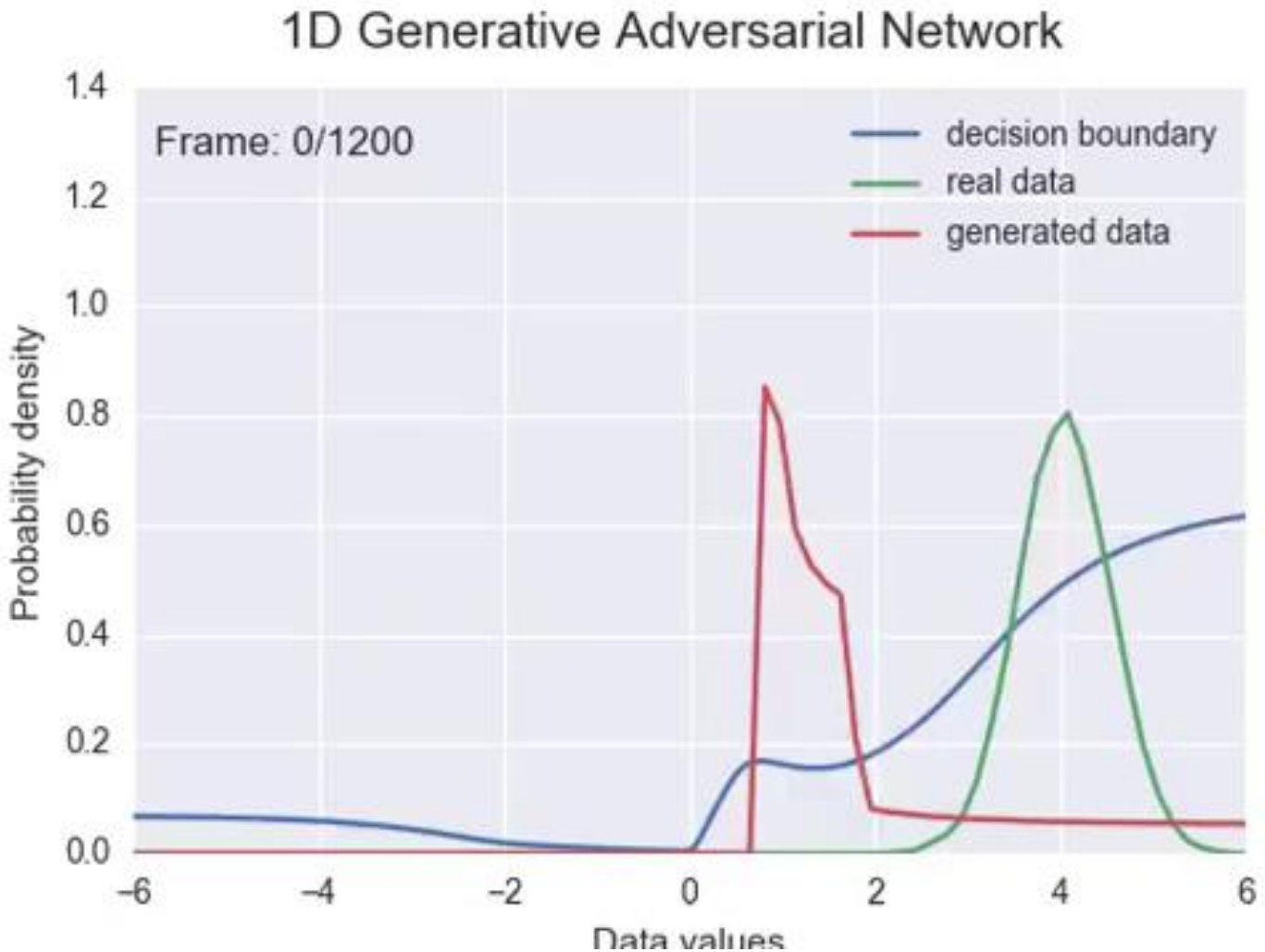
  end for

  - 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:

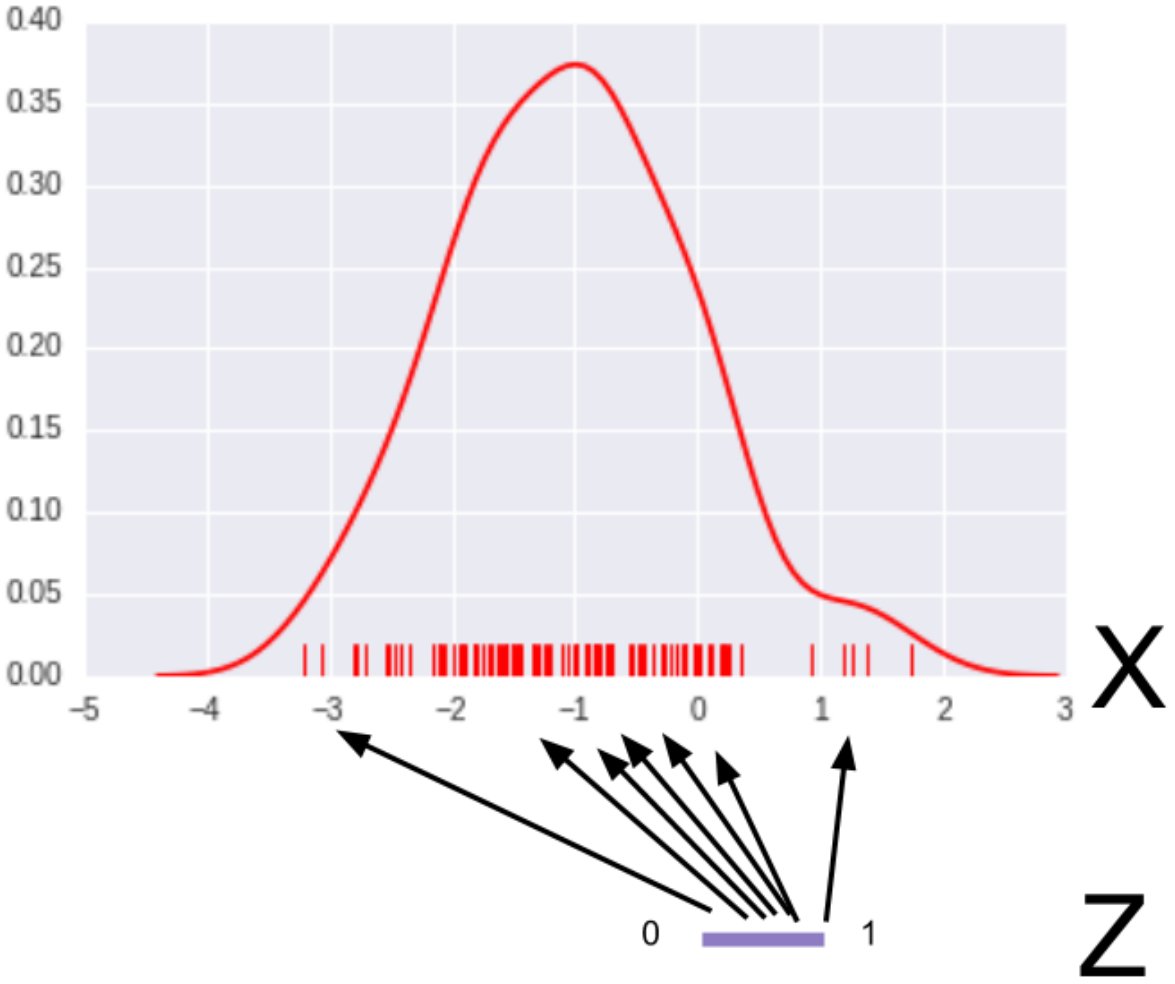
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

GAN training – 1D experiment



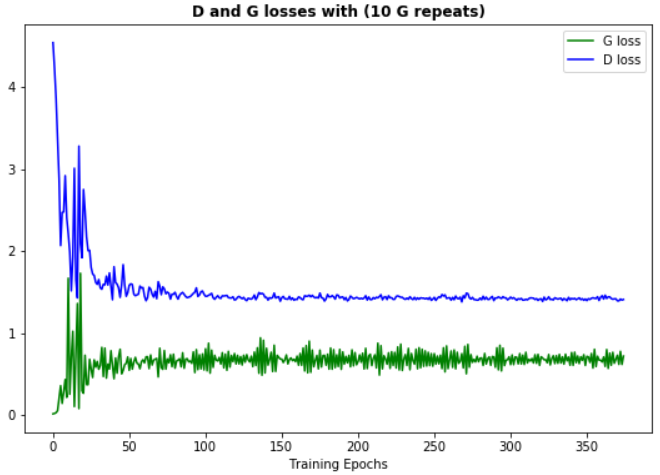
GAN training – 1D experiment



GAN Training – Loss Function

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

```
########### Loss calculations #########################
d_loss = -tf.reduce_mean(tf.log(D_legit) + tf.log(1. - D_fake))
g_loss = -tf.reduce_mean(tf.log(D_fake))
```



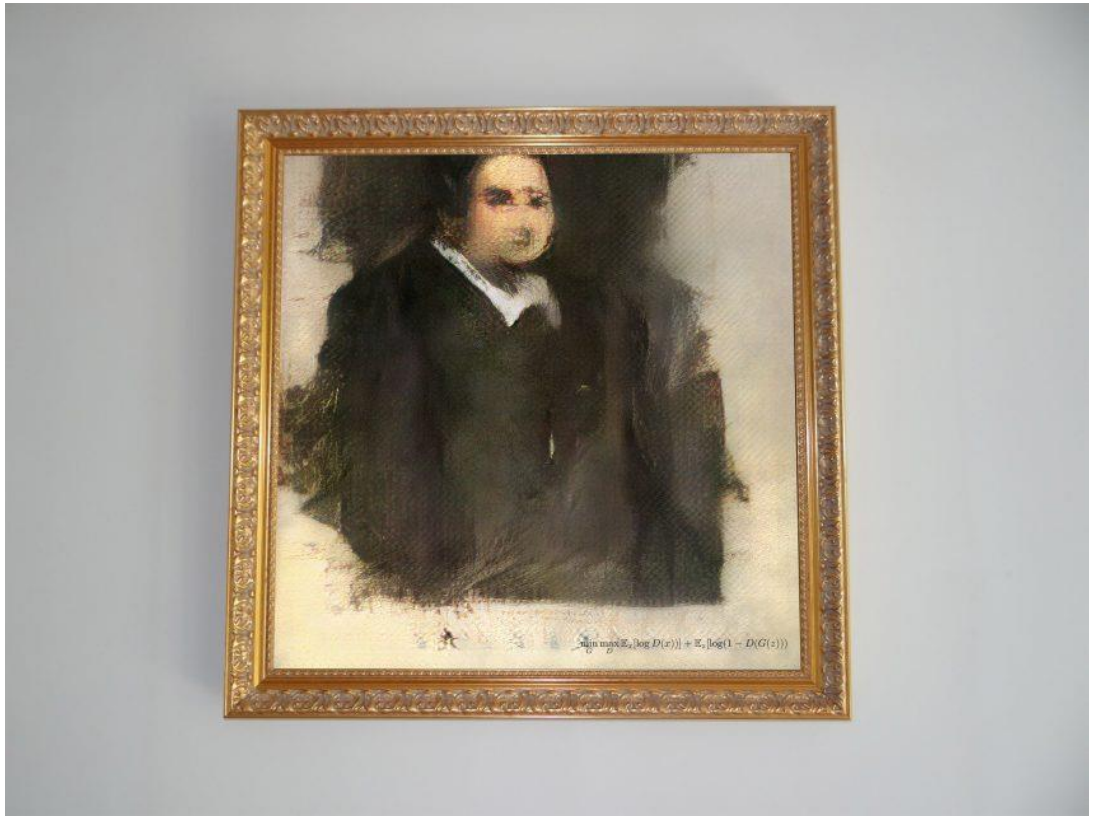

```
d_loss = tf.nn.sigmoid_cross_entropy_with_logits(logits=d_no_sigmoid_output_x, labels=tf.ones_like(d_output_x)) +
         tf.nn.sigmoid_cross_entropy_with_logits(logits=d_no_sigmoid_output_z, labels=tf.zeros_like(d_output_z))

g_loss = tf.nn.sigmoid_cross_entropy_with_logits(logits=d_no_sigmoid_output_z, labels=tf.ones_like(d_output_z))
```

GAN Applications on Image Data

- CelebA: Generating celebrity faces
  - Progressive Growing of GANs for Improved Quality, Stability, and Variation - Nvidia Research

First AI-Generated Painting Sold (Forbes - 29.10.2018)

Today at Christie's New York, the first AI-generated painting sold for $432,500 - it was expected to sell for $10,000. A type of AI algorithm known as *Generative Adversarial Networks (GAN)* and trained by a group called Obvious art generated the work.



Challenges

GANs are, at the moment, very unstable and need many tricks to converge. Also they are notoriously difficult to train. Without the right hyperparameters, network architecture, and training procedure, there is a high chance that either the generator or discriminator will overpower the other.

- If discriminator learns faster, it can detect generated data as fake easily and generator may never get better because all the data it generates will be labeled as fake.

- If generator learns faster, discriminator fails to distinguish real and fake data so system never generates realistic data.

Need to tune all parameters for 2 or more different networks to make them work together.

Mode collapse

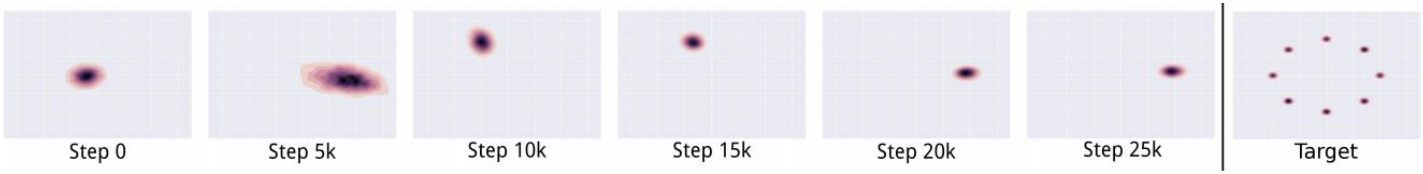
- With a theoretical point of view,
  - Generator doesn’t aim to produce realistic outputs
  - It aims to make the Discriminator label generator output as real
  - If discriminator labels the Generator output as real, Generator is successfull
  - Generator always tries to fool the Discriminator

- Mostly, unstability causes mode collapse
  - Most severe form of non-convergence.
  - Generating same data always.
  - Overfitting to a single category.
  - It is like getting stuck on a local maxima.
  - Trained with MNIST dataset, generates 1 always.

Mode collapse

Oscillation

GANs can train for a very long time, generating very many different categories of samples, without clearly generating better samples.

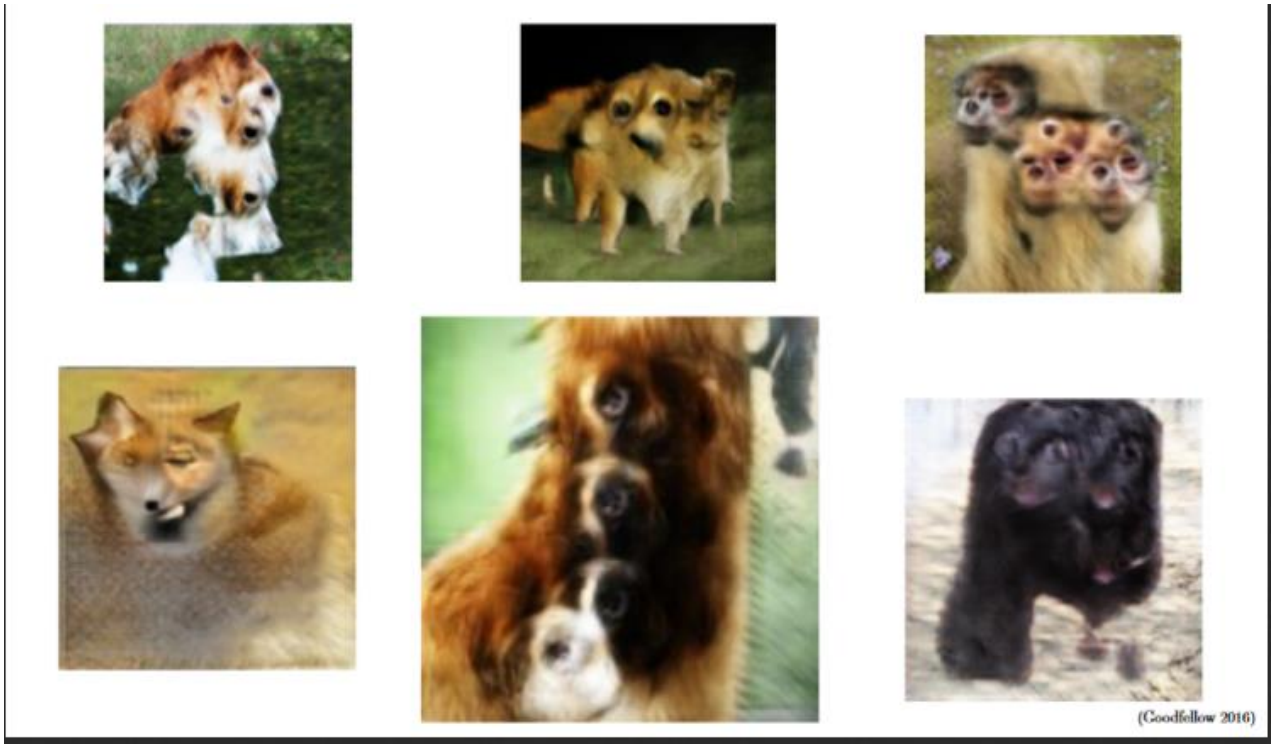


Minibatch Discrimination

- Discriminator evaluates samples individually.
- Use all samples in a batch together for evaluation.
- Similarity between samples is a new parameter for evaluation.
- Prevents generating similar samples, thus prevents mode collapse.

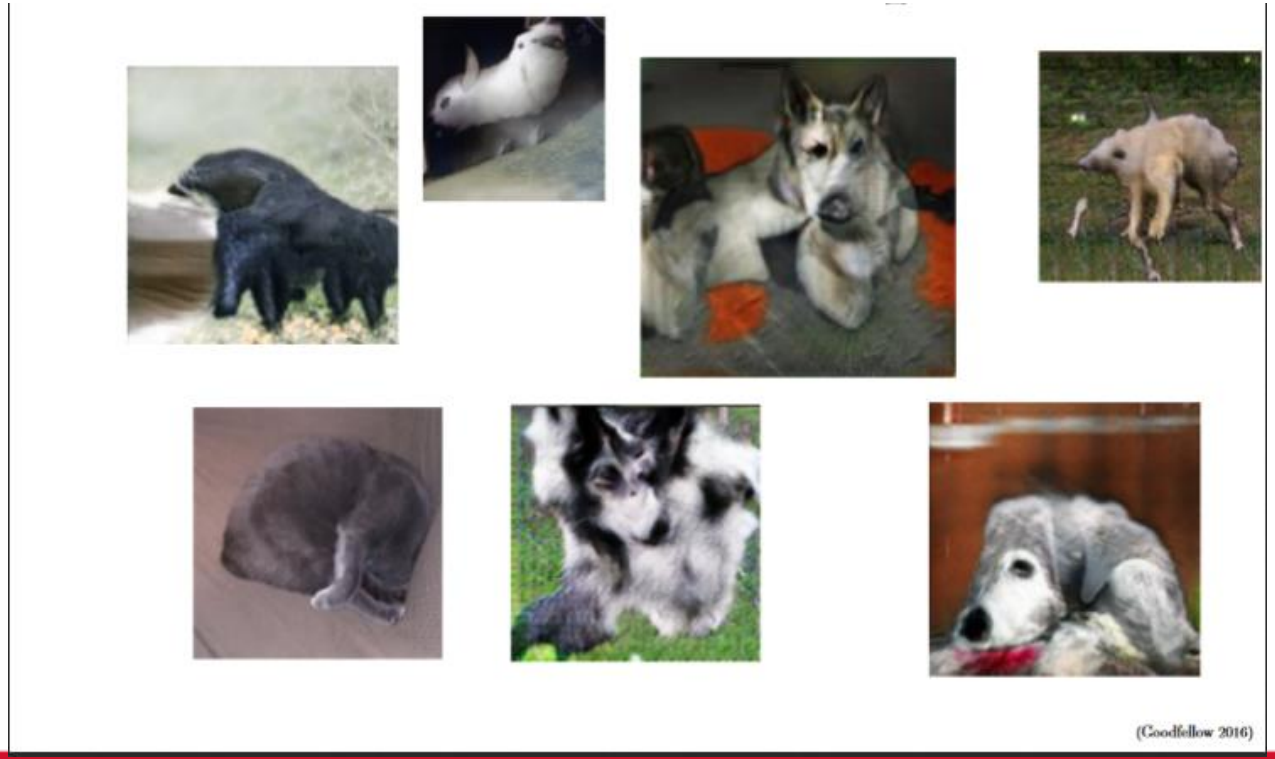
Problems with counting

GANs fail to differentiate how many of a particular object should occur at a location. As we can see below, it gives more number of eyes in the head than naturally present.



Problems with perspective

GANs fail to adapt to 3D objects. It doesn't understand perspective, i.e. difference between frontview and backview. As we can see below, it gives flat (2D) representation of 3D objects.



Problems with global structure

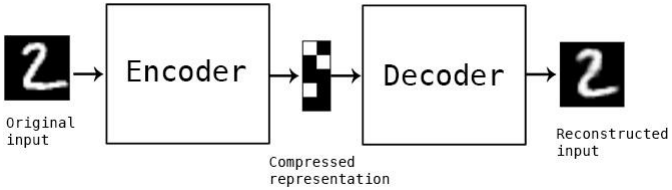
Same as the problem with perspective, GANs do not understand a holistic structure. For example, in the bottom left image, it gives a generated image of a quadruple cow, i.e. a cow standing on its hind legs and simultaneously on all four legs. That is definitely not possible in real life!

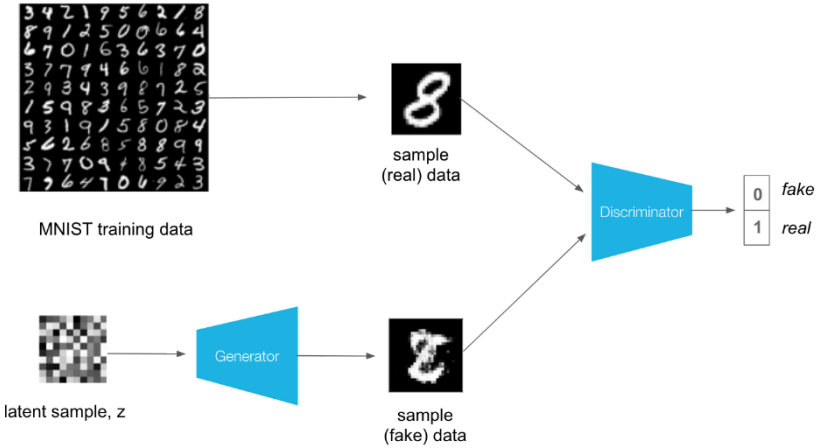
GAN metrics

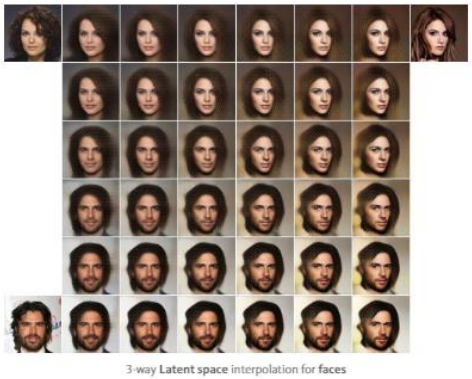
Human perception : People decide which is better

GAN metrics

Latent space interpolation : Smooth transitions between generated samples when input noise is interpolated over the latent space.






3-way **Latent space** interpolation for **faces**

GAN metrics

Inception Score :

1 - Every realistic image should be recognizable, which means that the score distribution for it must be, ideally, dominated by one class. (Easily classified)

2- Class distribution over the whole sample should be as close to uniform as possible, in other words, a good generator is a diverse generator. (Samples are from different classes)

Uses InceptionNet to evaluate generated images. (Higher is better , SotA 52.52)

$$IS(x) = \exp(\mathbb{E}_x \left[KL\left(p(y|x) || p(y) \right) \right])$$

Fréchet Inception Distance : Distance between generated and real sample distributions (lower is better , SotA 18.65)

Large Scale GAN Training for High Fidelity Natural Image Synthesis

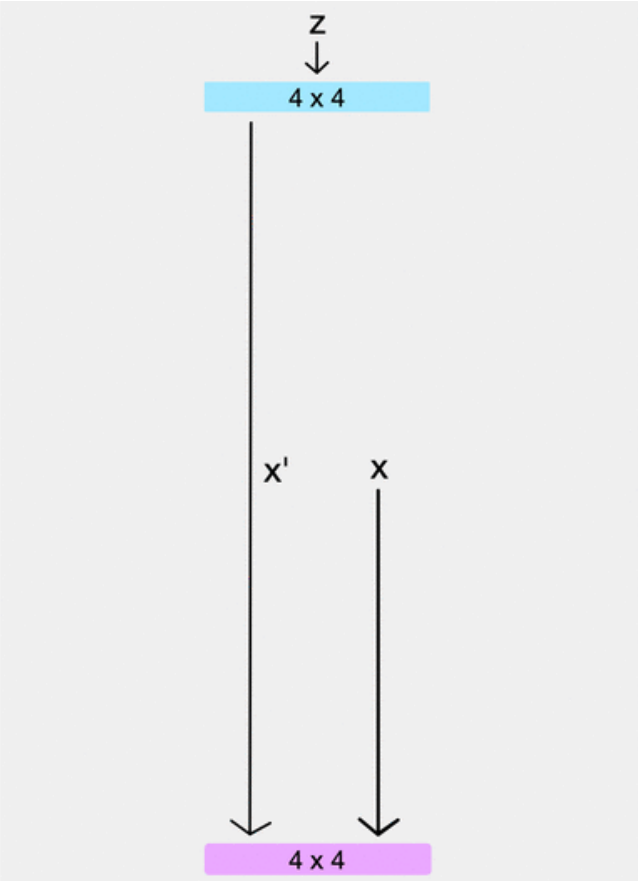
- BigGAN – 28.09.2018
- 512x512 resolution
- IS = 186 (previous 52) FID = 9 (previous 18)

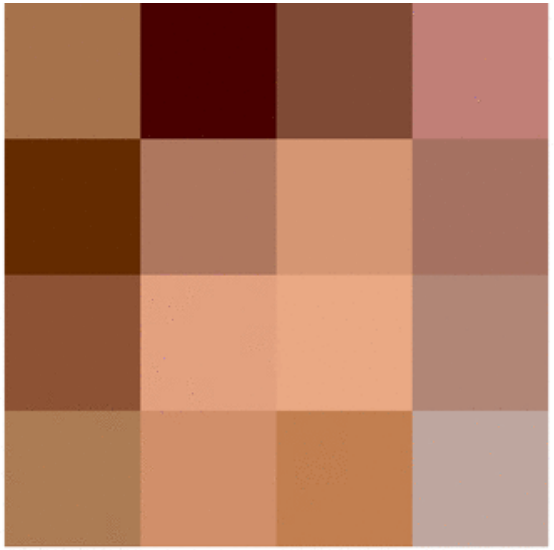
Large Scale GAN Training for High Fidelity Natural Image Synthesis

- JFT-300M dataset
  - Google's internal dataset, not public
  - 512x512 3 channel resolution 300 million images 18000 classes
- 48 hours of training on 512 TPUv3
- Google DeepMind
- New State of the Art for GANs

GAN Applications on Image Data

- CelebA: Generating celebrity faces
  - Progressive Growing of GANs for Improved Quality, Stability, and Variation - Nvidia Research





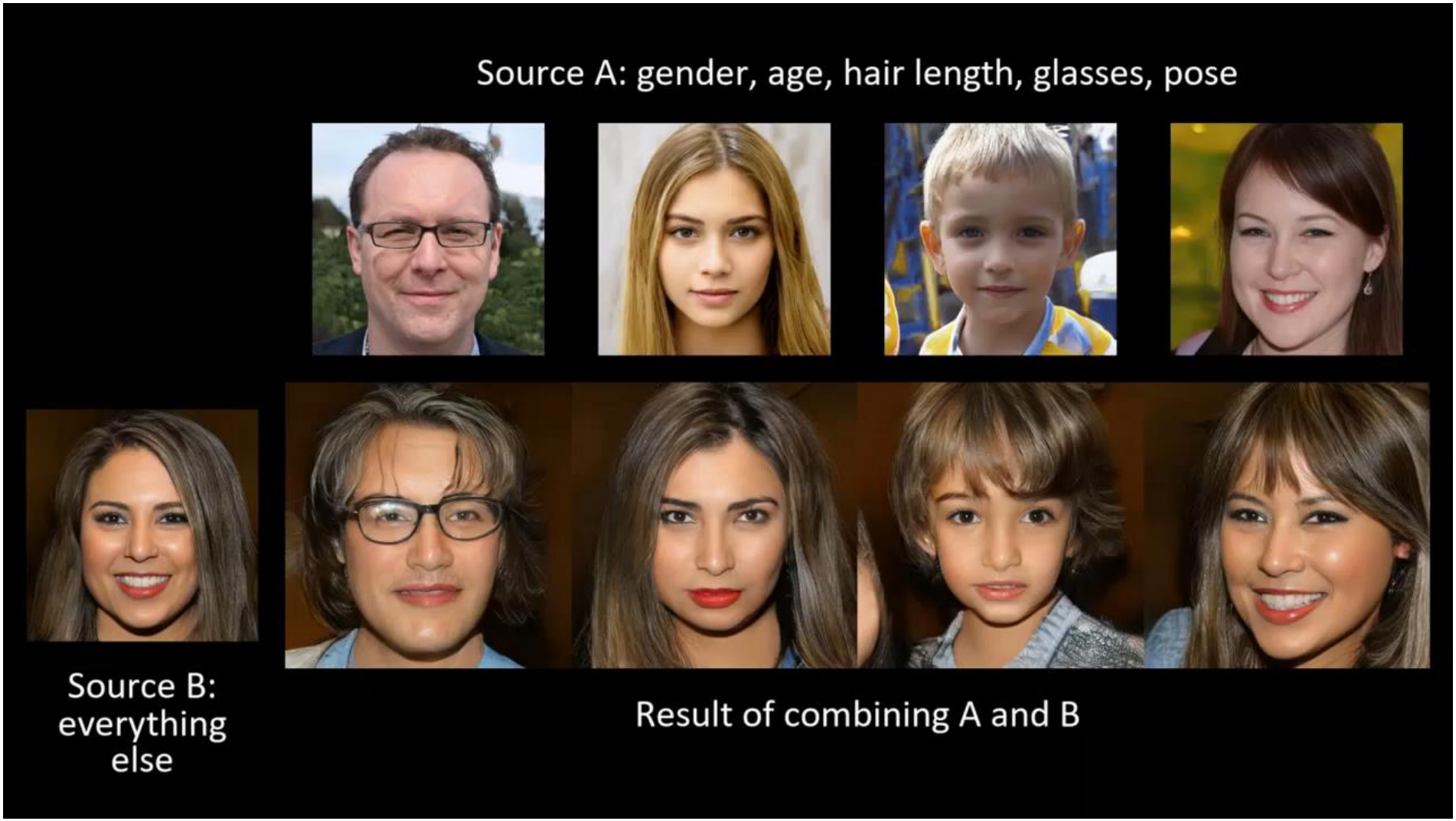
Training time: 0 days
4x4 resolution

Generator
Discriminator

z = random code
x = real image
x' = generated image

A Style-Based Generator Architecture for Generative Adversarial Networks (12.12.2018)

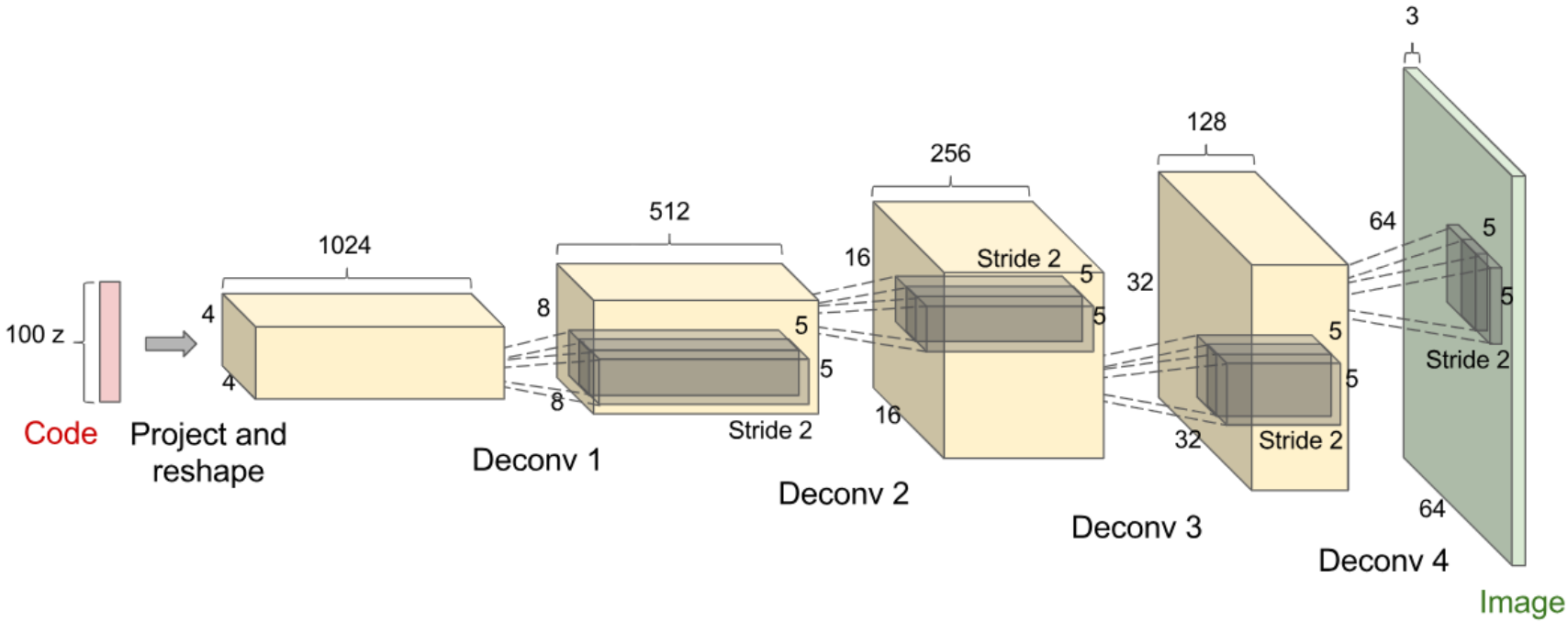
A Style-Based Generator Architecture for Generative Adversarial Networks



Deep Convolutional GAN - DCGAN

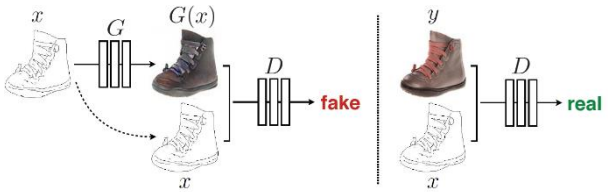
CNNs for Discriminator

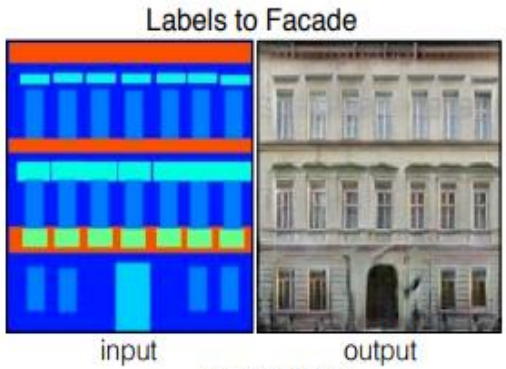
Transposed Convolution (Deconvolution) for Generator

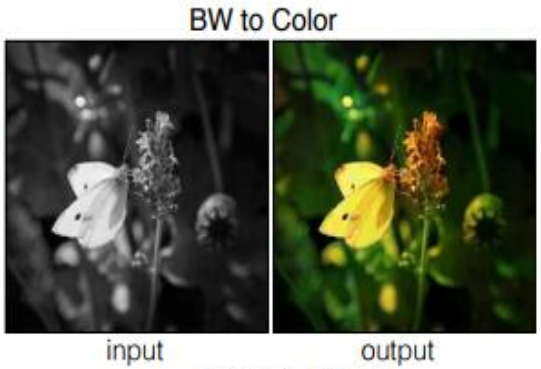


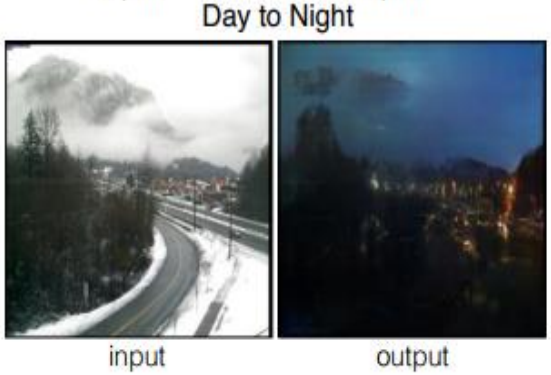
GAN Applications on Image Data

- pix2pix: A general purpose solution to image-to-image translation









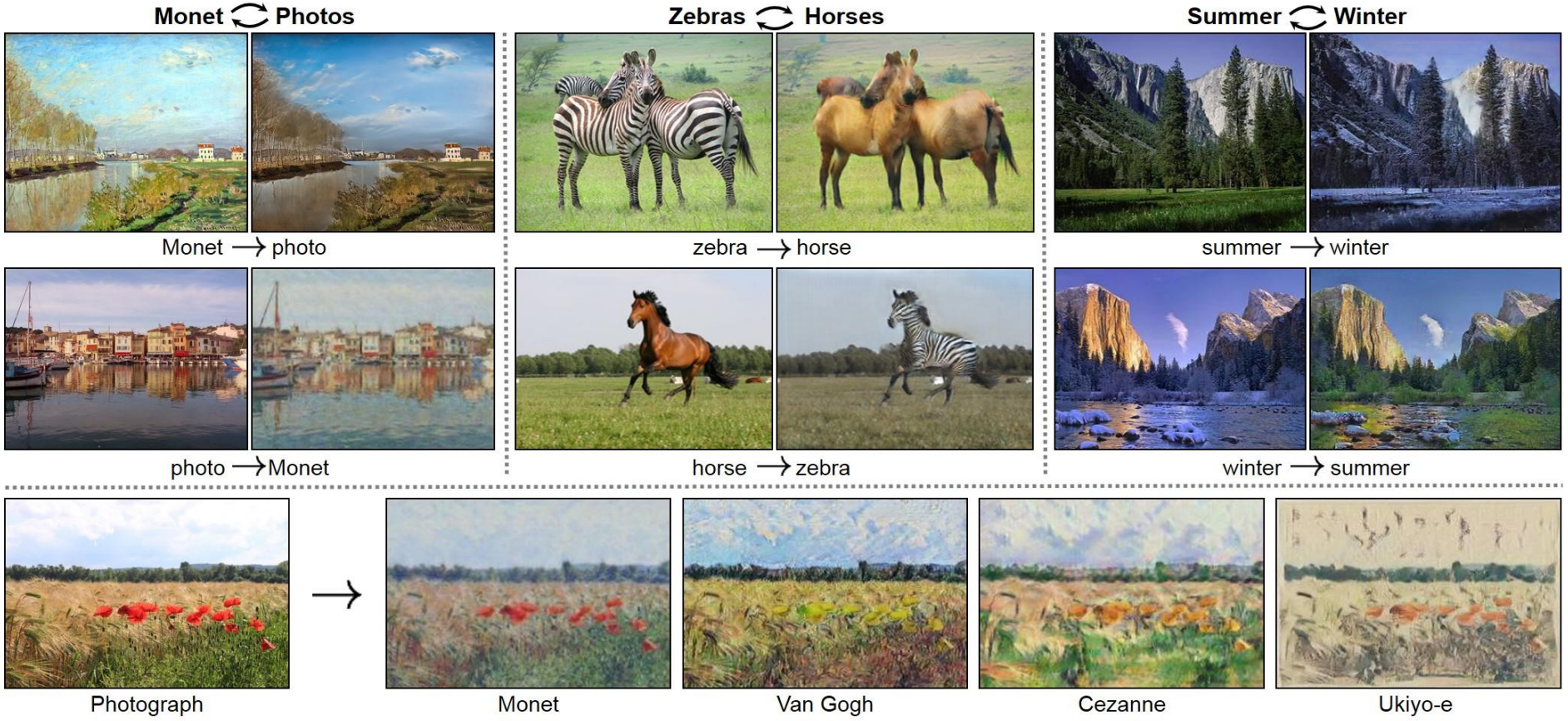
GAN Applications on Image Data

GAN Applications on Image Data

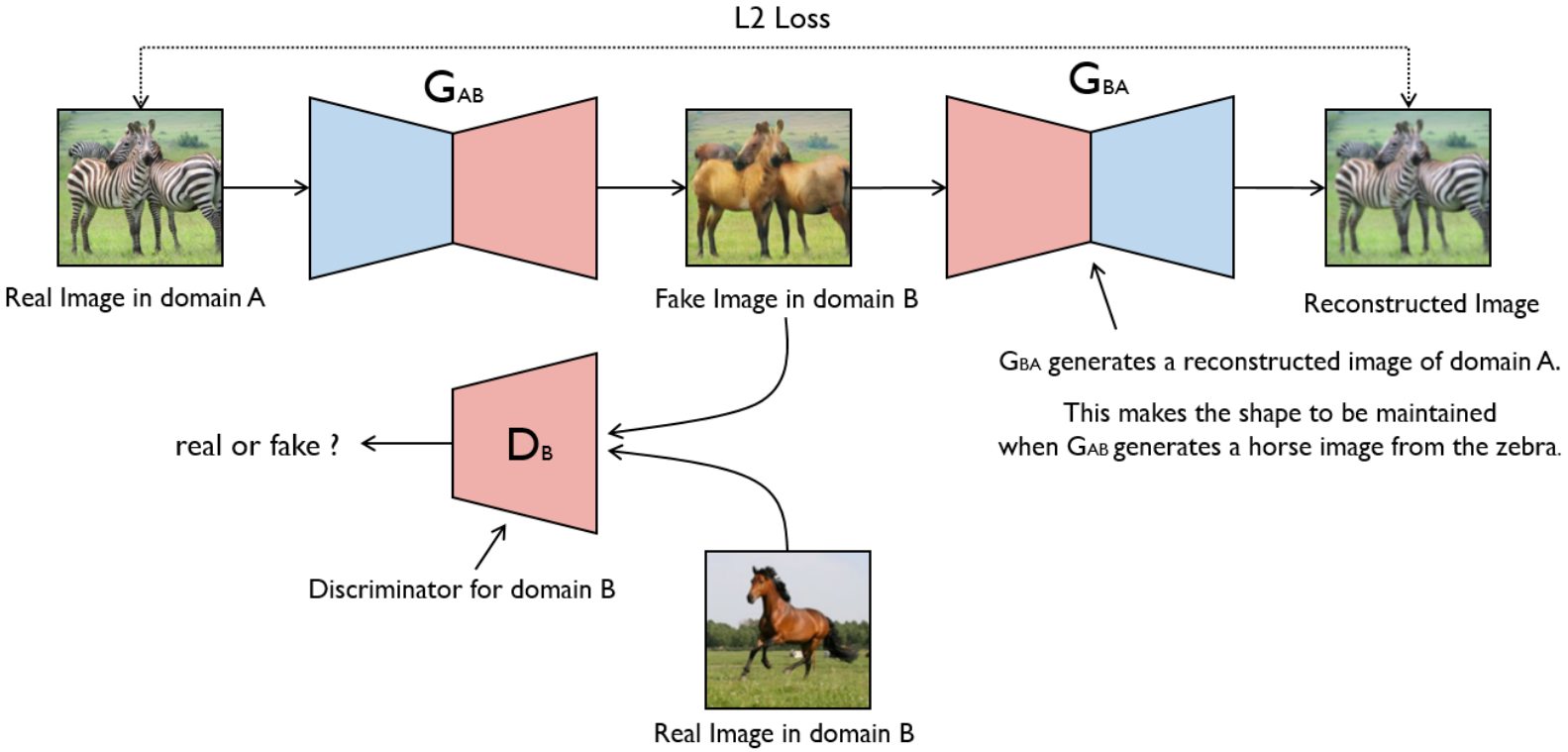
- Style transfer for scene images of cities and wide area
  - Time of the day, weather, season and artistic edit.
  - Deep Photo Style Transfer - CycleGAN

GAN Applications on Image Data

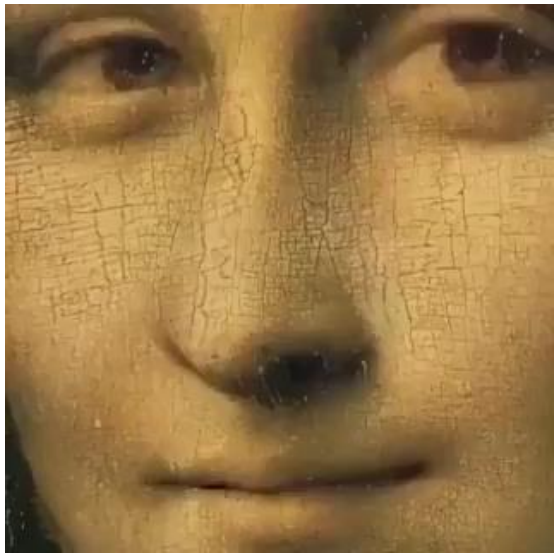
- Style transfer for scene images of cities and wide area
  - Time of the day, weather, season and artistic edit.
  - Deep Photo Style Transfer - CycleGAN



CycleGAN

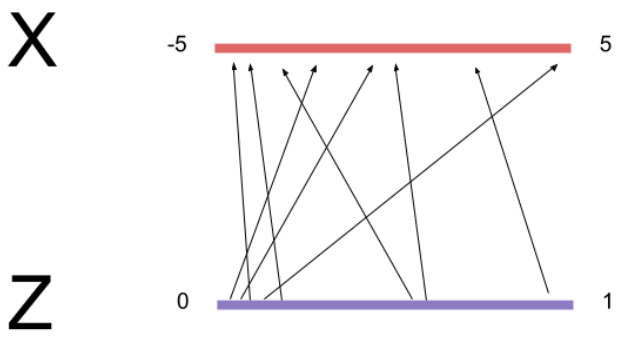

L2 Loss
G_{AB}
G_{BA}
Real Image in domain A
Fake Image in domain B
Reconstructed Image
G_{BA} generates a reconstructed image of domain A.
This makes the shape to be maintained
when G_{AB} generates a horse image from the zebra.
real or fake ?
D_B
Discriminator for domain B
Real Image in domain B

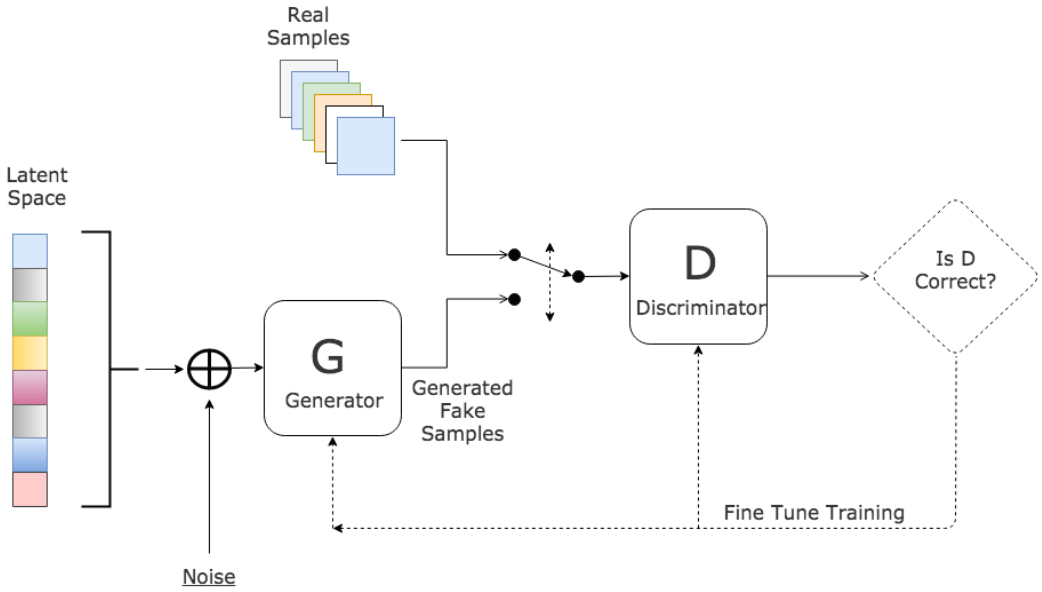
DeepFakes



Entanglement Problem

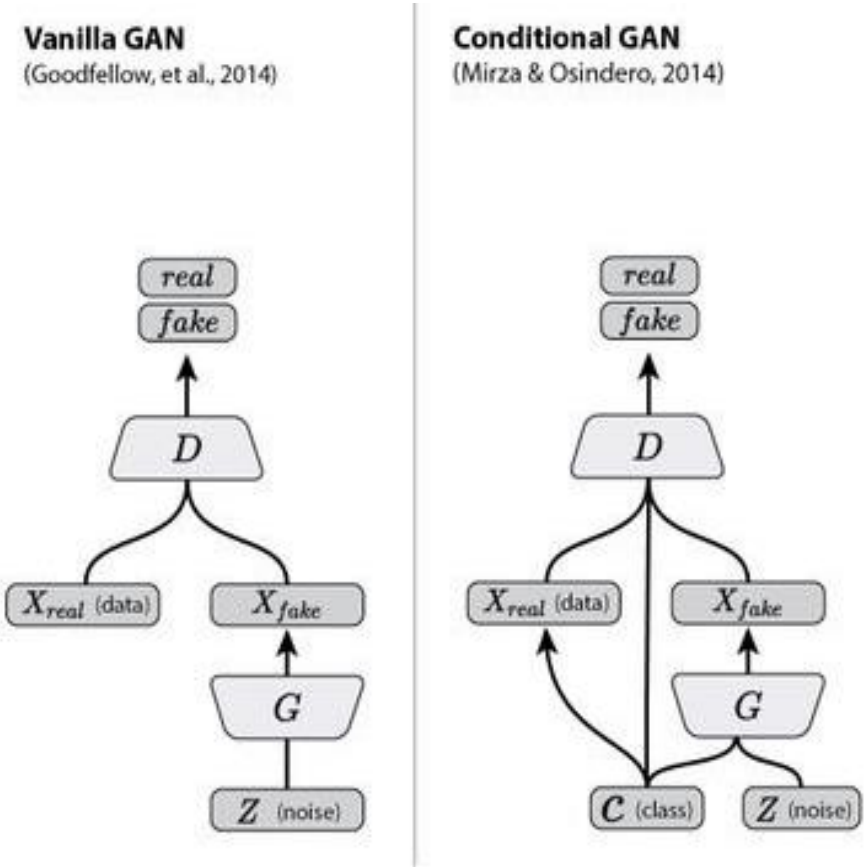
- Only way to change the output is to change the input noise vector Z
- To generate a sample with desired attributes (number 7, bold digit, a man with glasses etc.) how do you change the noise?
- Which value of input vector represent which attribute?
- Mostly none of them directly represents a meaningful attribute (boldness or glasses) because the representations are entangled

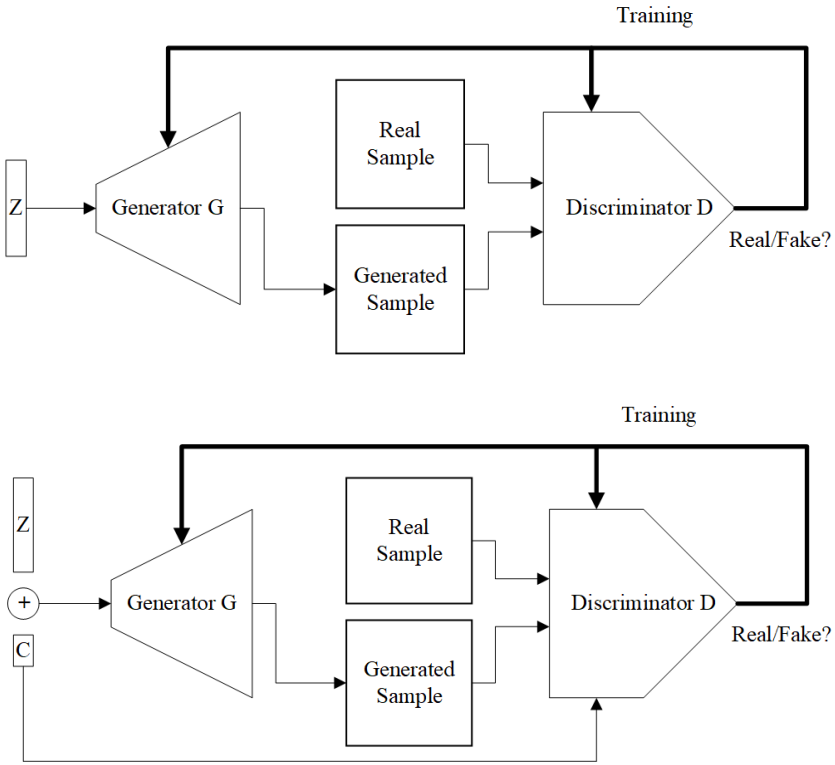




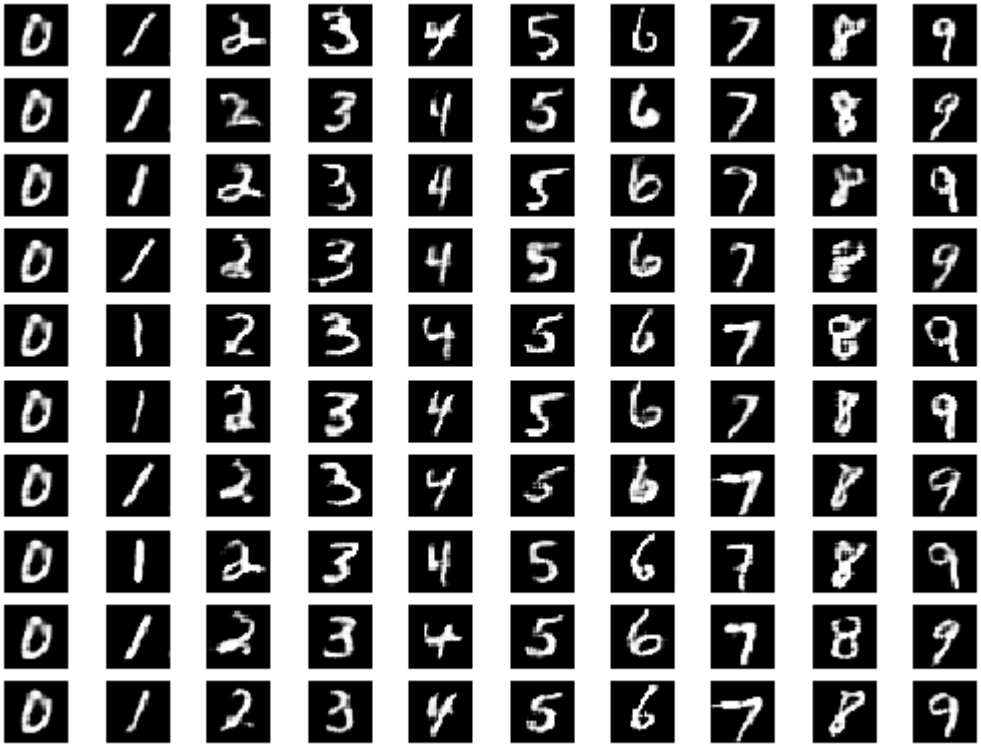
Conditional GAN

- Adds condition parameter to GAN to control the class of determined sample



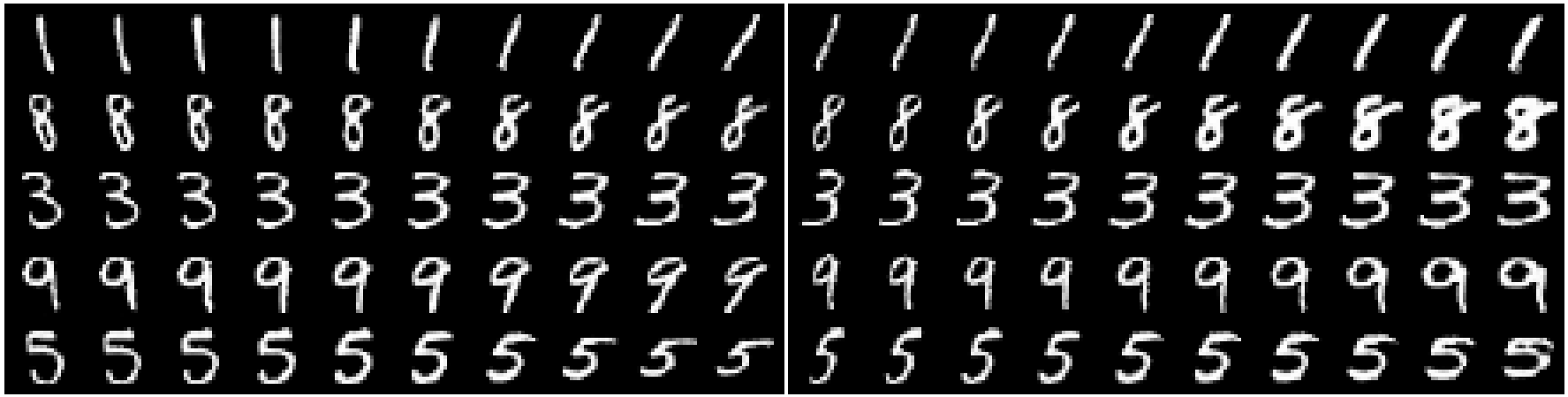


Conditional GAN



InfoGAN

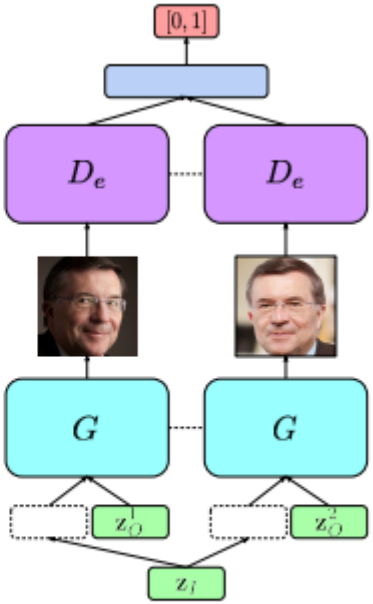
- Chen et al. InfoGAN [8] : proposes a solution to entanglement problem
- Generator uses input noise vector ,there is no restrictions in which the generator may use this noise.
- It is possible that the noise will be used by the generator in a highly entangled way, causing the input vector z to not correspond to semantic features of the output data.
- A simple modification to the generative adversarial network objective that encourages it to learn interpretable and meaningful representations.
- Combination of noise z, latent code c : G(z,c)



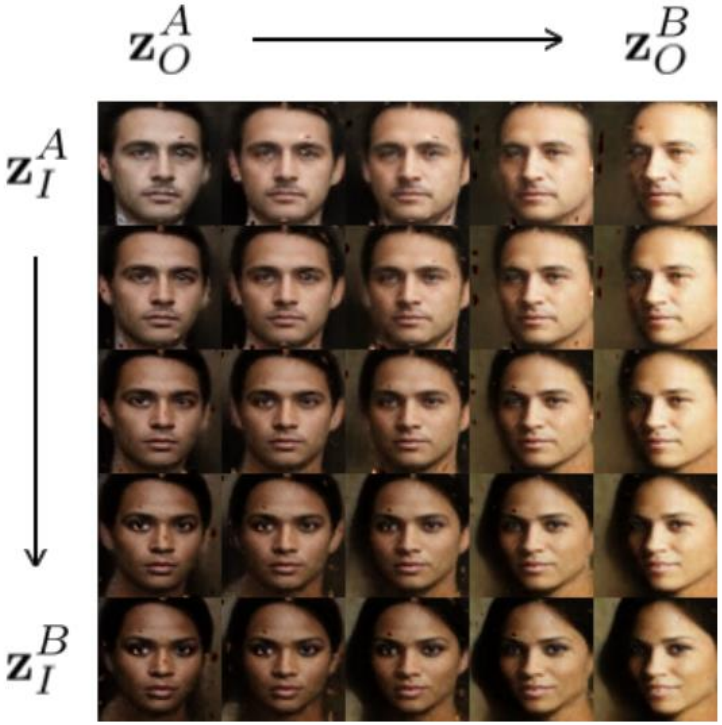
Siamese GAN

- Identical GANs
- Pairwise training
- Weight sharing
- Identity code and attribute code

(a) DCGAN

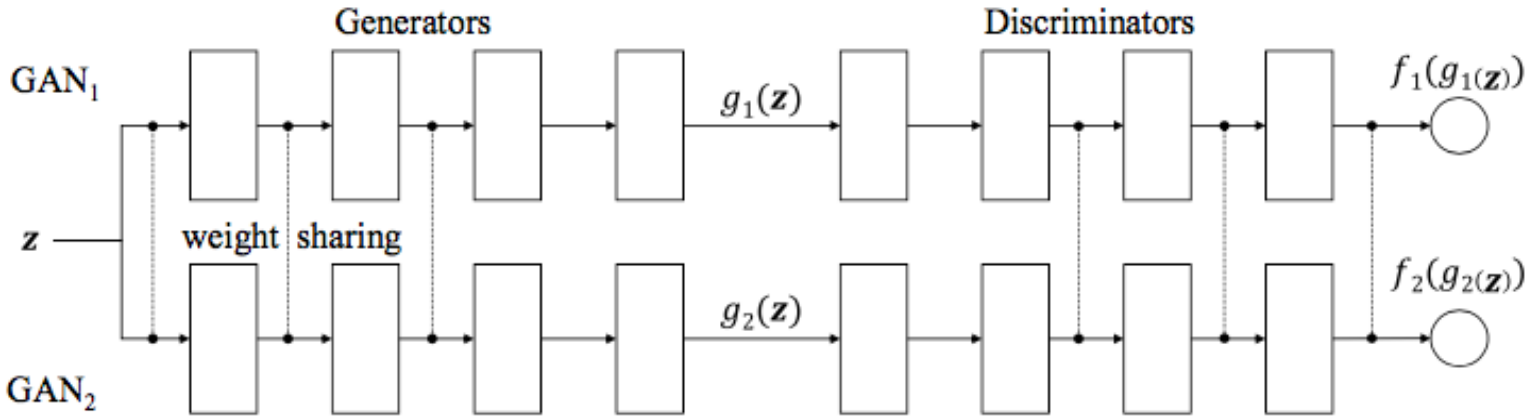


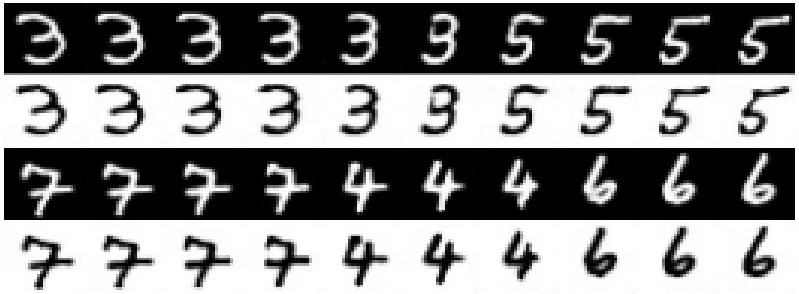
(b) SD-DCGAN



Coupled GAN

- 2 identical GANs
- Partial weight sharing
- Same high abstraction layers





Wasserstein GAN

- Different loss function for better training

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -\log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_{w} \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right) \right]$	$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} -f\left(G\left(z^{(i)}\right)\right)$

Improved Techniques for Training GANs

- feature matching
- minibatch discrimination
- historical averaging
- one-sided label smoothing
- virtual batch normalization

GAN Zoo – Different GAN types
https://github.com/hindupuravinash/the-gan-zoo