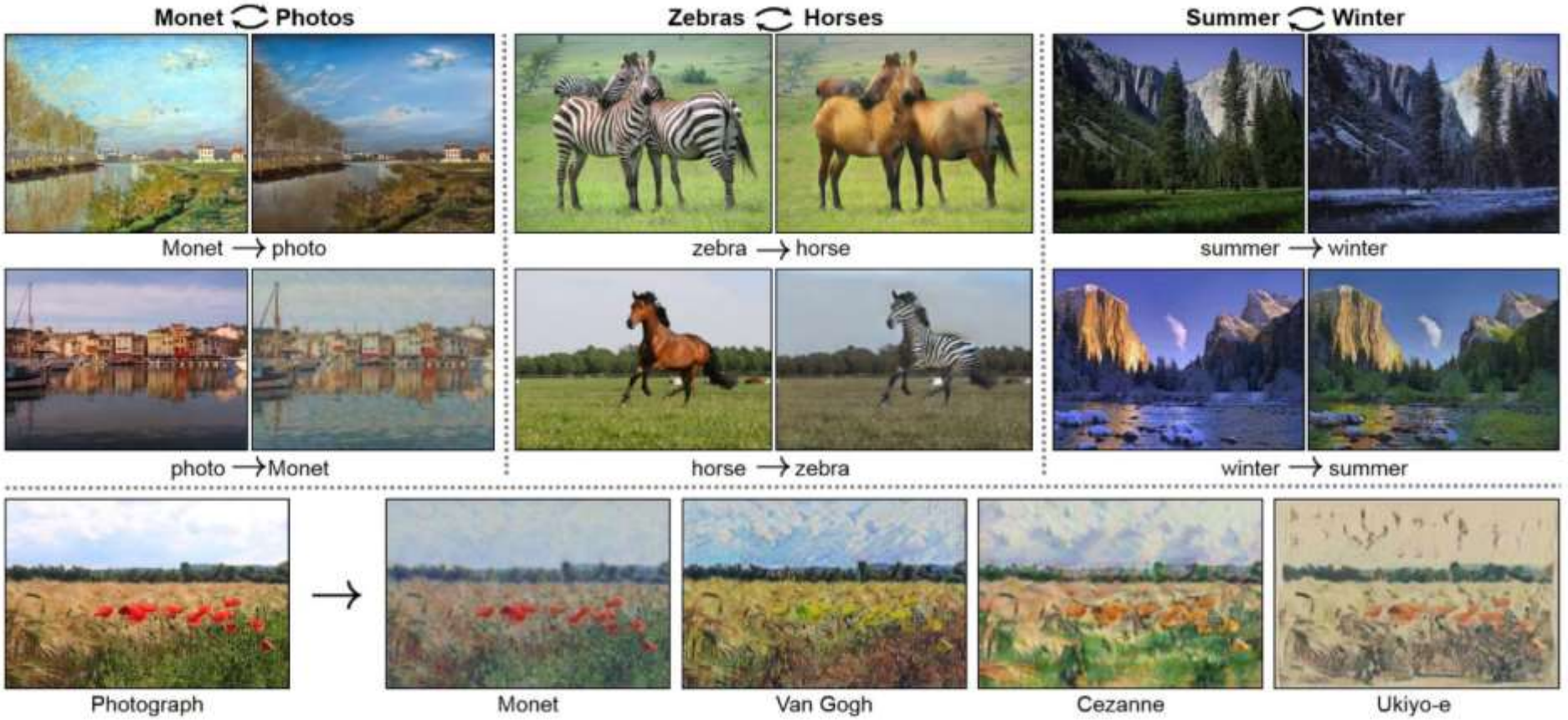
Lecture 16: Deep Generative Models V: Variants of GANs

Lan Xu
SIST, ShanghaiTech
Fall, 2023

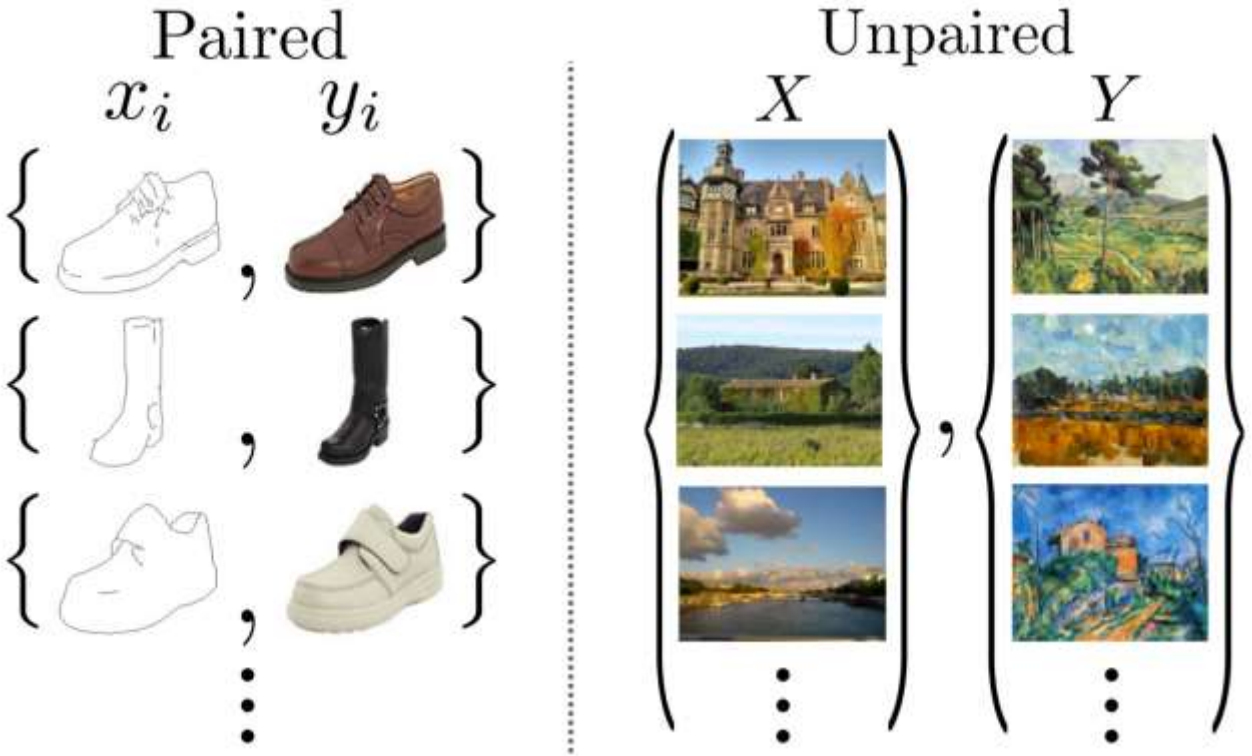
CycleGAN

- Image-to-image translation without paired data



CycleGAN

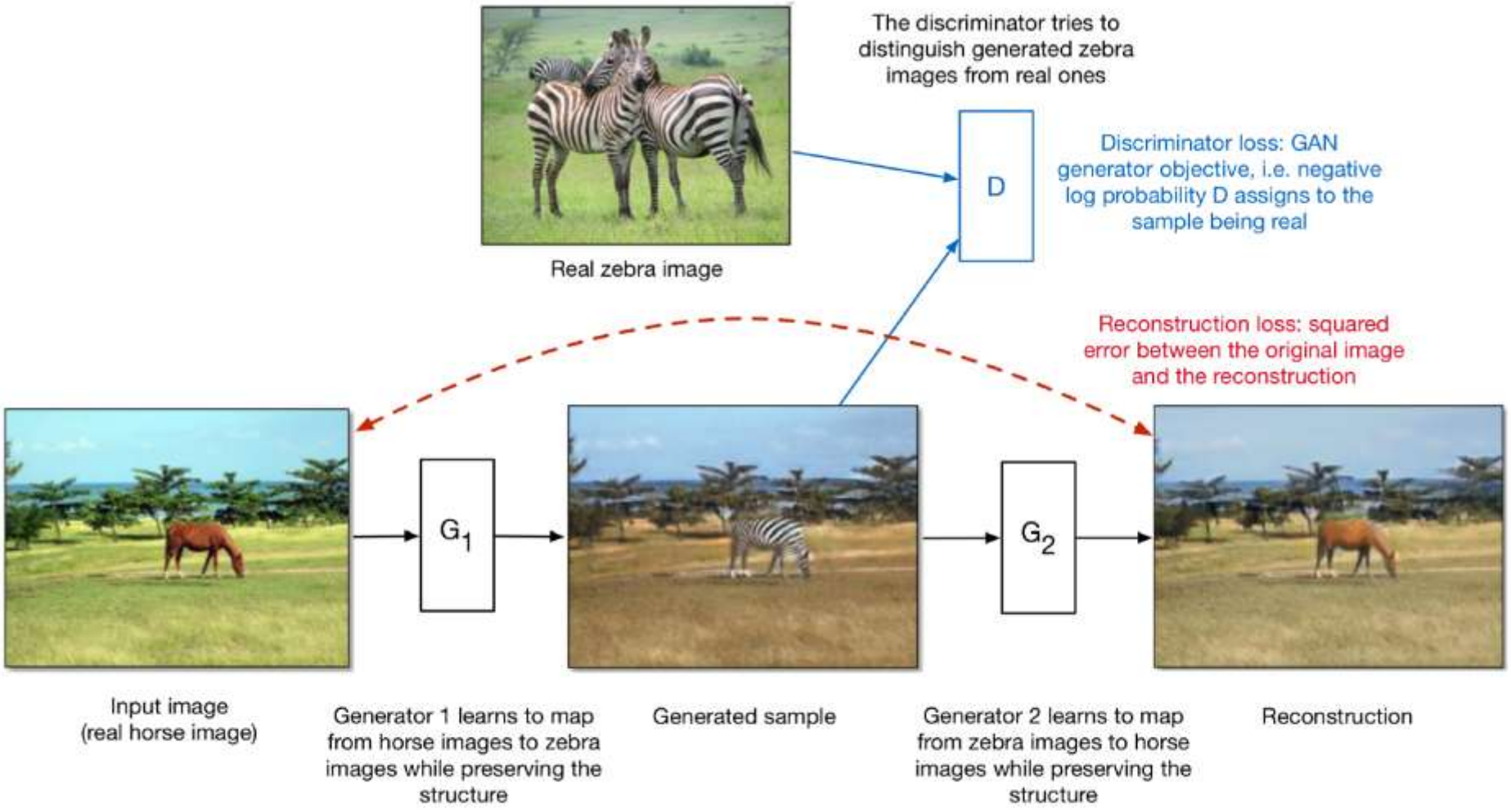
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.



CycleGAN

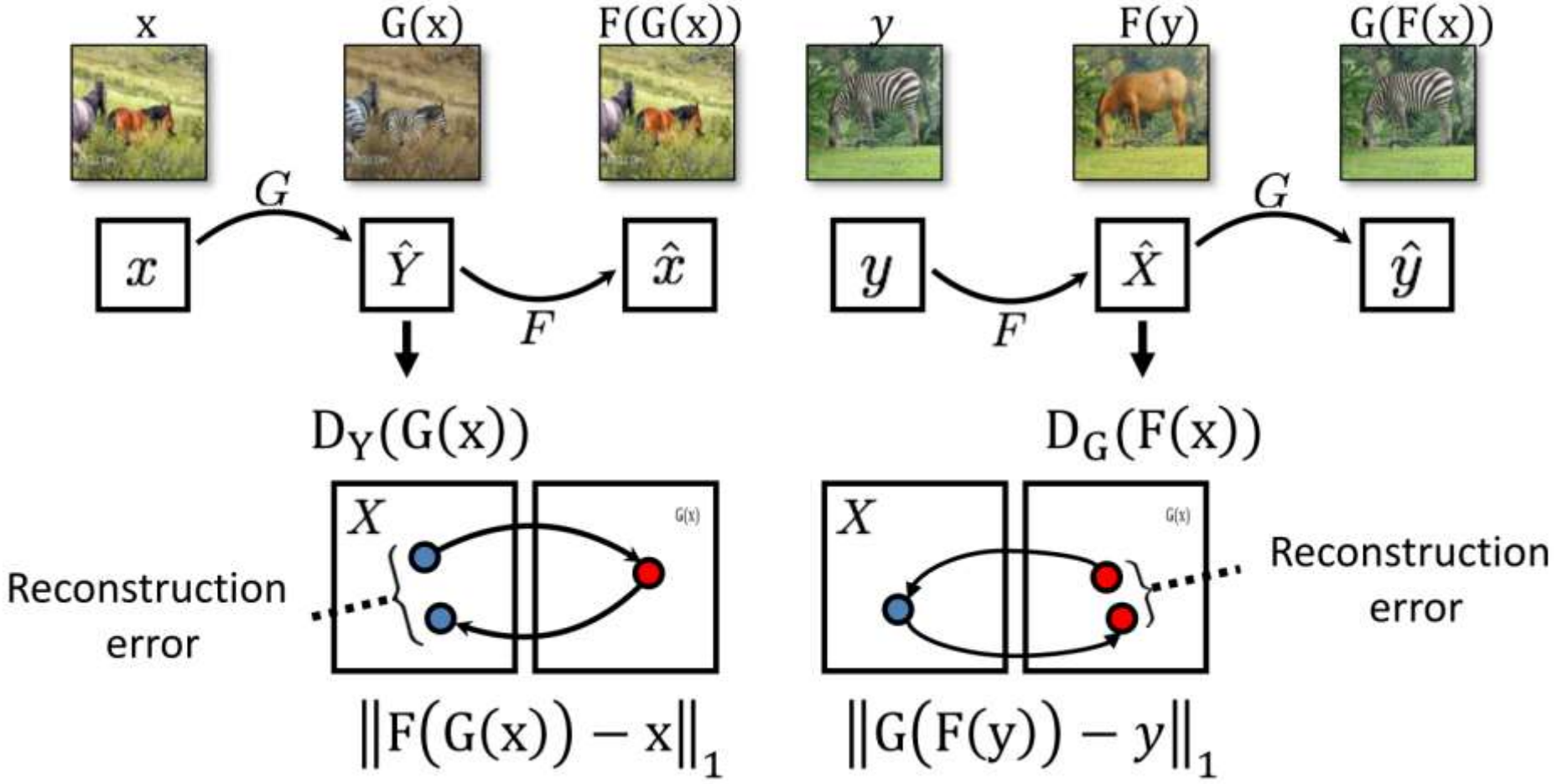
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - Make sure the generators are **cycle-consistent**: mapping from style 1 to style 2 and back again should give you almost the original image.

CycleGAN



Total loss = discriminator loss + reconstruction loss

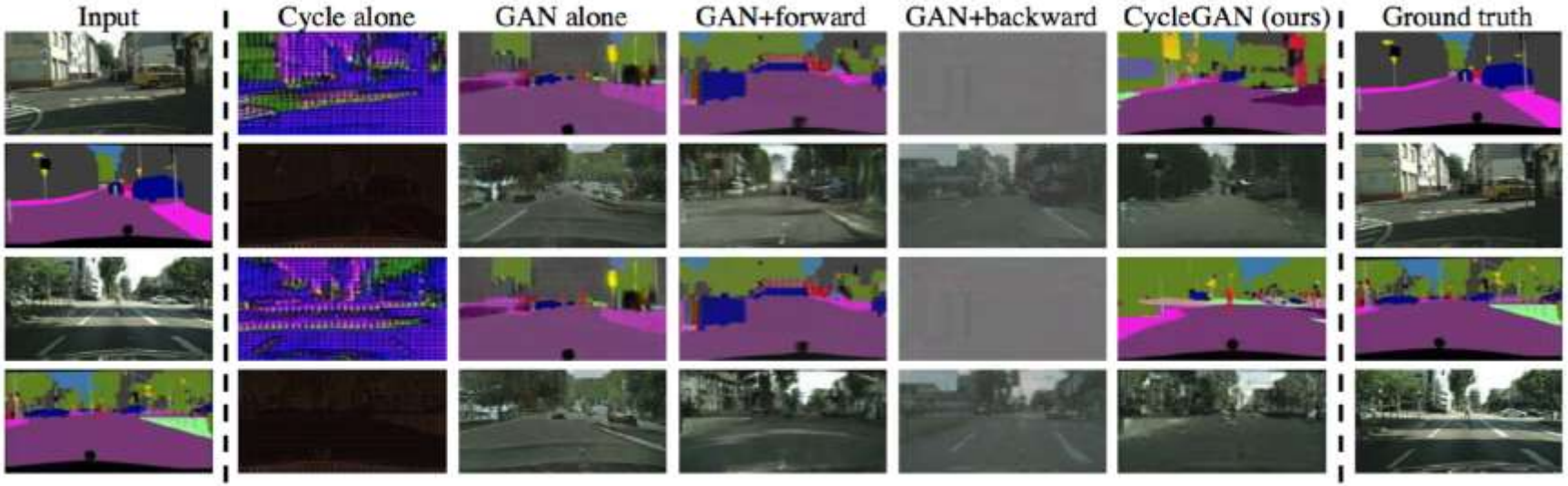
CycleGAN


x
G(x)
F(G(x))
y
F(y)
G(F(x))
G
F
G
F
x
$\hat{Y}$
$\hat{x}$
y
$\hat{X}$
$\hat{y}$
$D_Y(G(x))$
$D_G(F(x))$
X
G(x)
X
Reconstruction error
Reconstruction error
$\|F(G(x)) - x\|_1$
$\|G(F(y)) - y\|_1$

CycleGAN

- Results

Style transfer between road scenes and semantic segmentations (labels of every pixel in an image by object category):



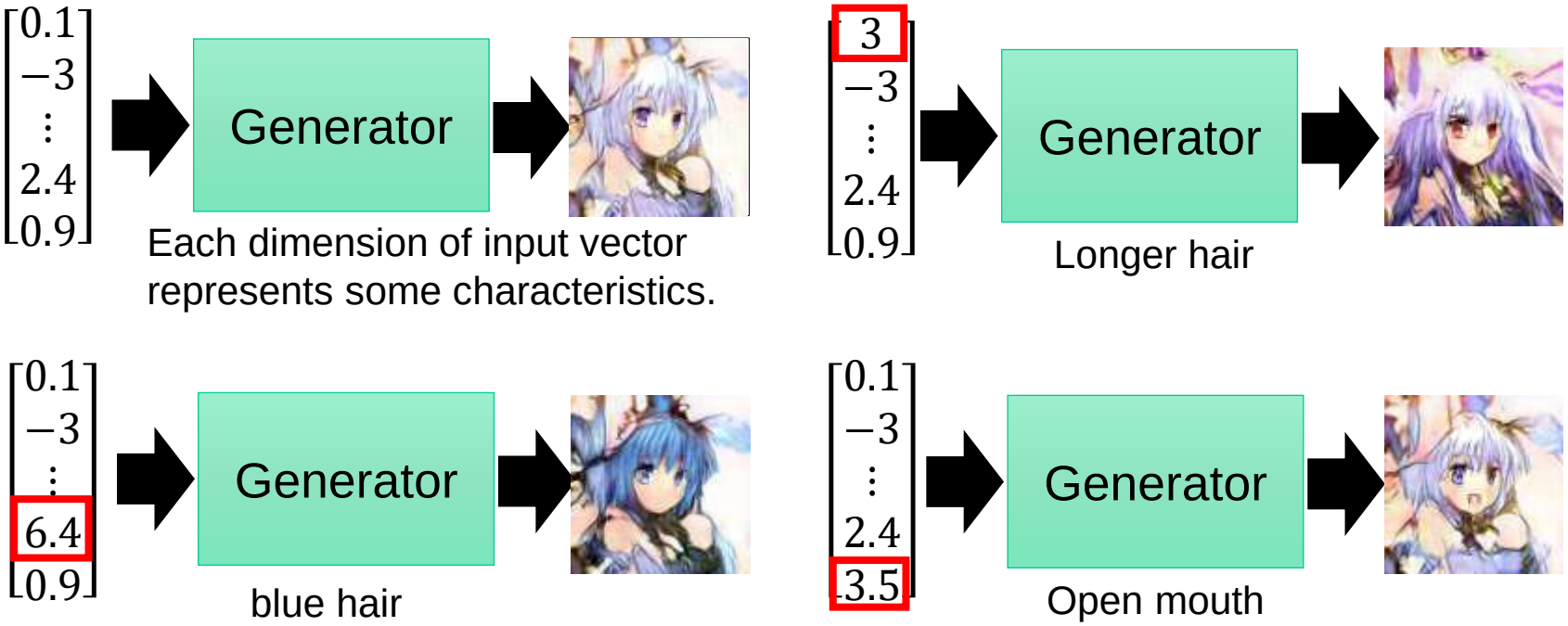
CycleGAN

- Results

- More details

https://hardikbansal.github.io/CycleGANBlog/

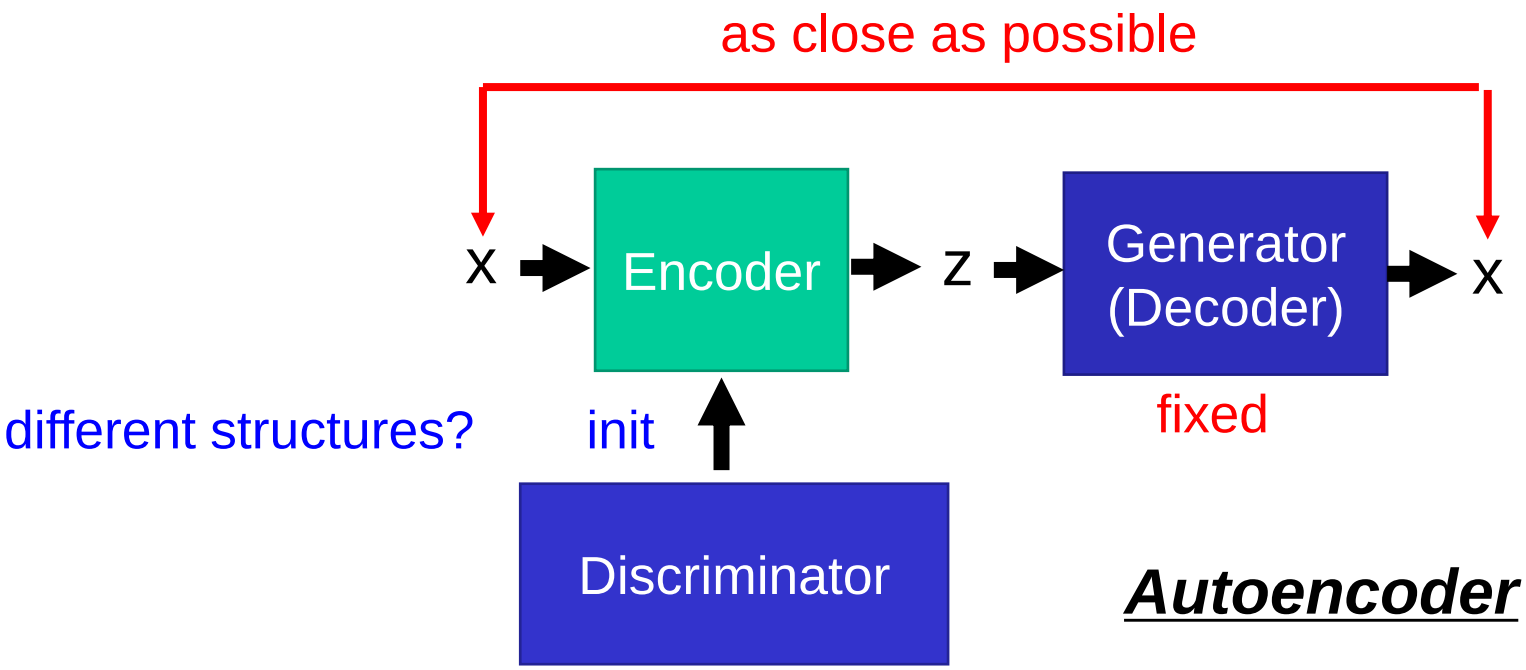
Modifying Input Code



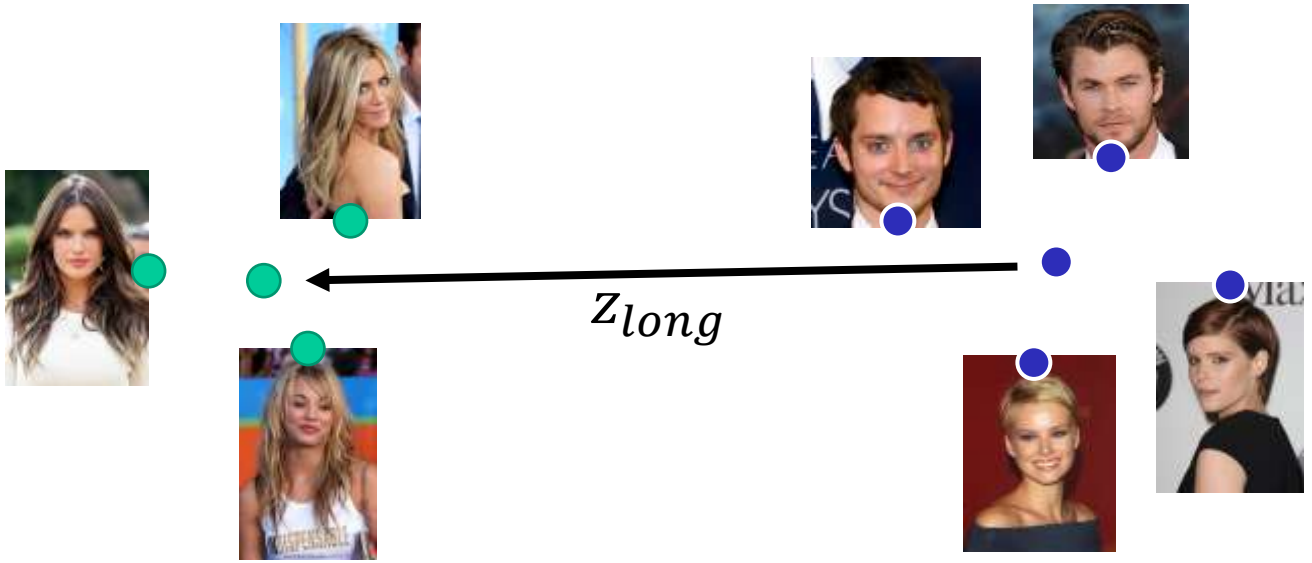
- The input code determines the generator output.
- Understand the meaning of each dimension to control the output.

GAN + Autoencoder

- We have a generator (input z, output x)
- However, given x, how can we find z?
  - Learn an encoder (input x, output z)



GAN + AE: attribute representation



$$z_{long} = \frac{1}{N_1}\sum_{x \in long} En(x) - \frac{1}{N_2}\sum_{x' \notin long} En(x')$$

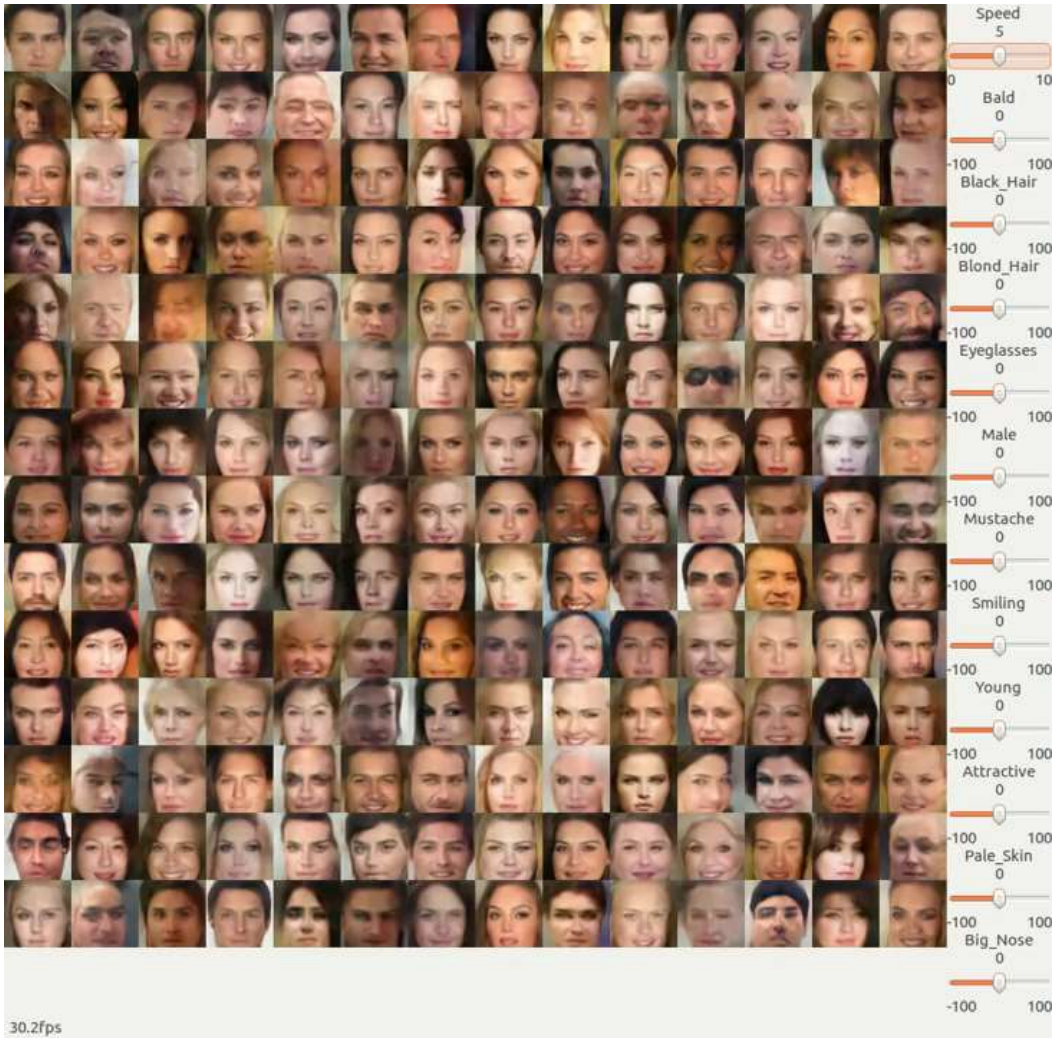
Short Hair

$$x \Rightarrow En(x) + z_{long} = z' \Rightarrow Gen(z')$$

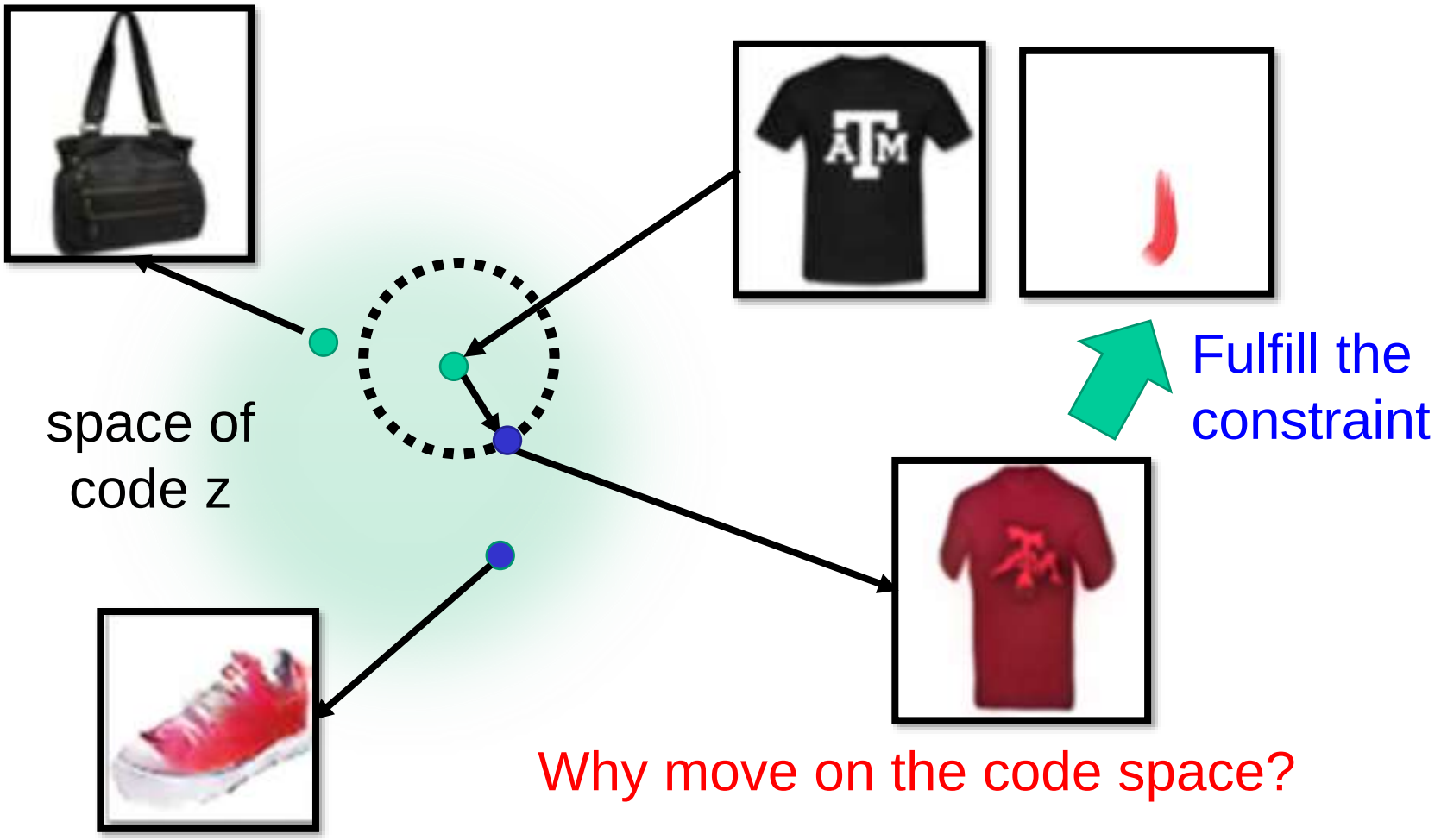
Long Hair

GAN + AE: attribute control

- Photo editing via attribute control

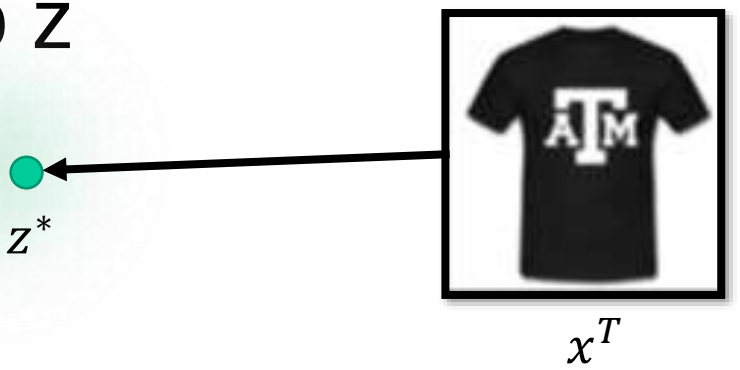


Attribute editing: basic idea



Back to z

- ***Method 1***



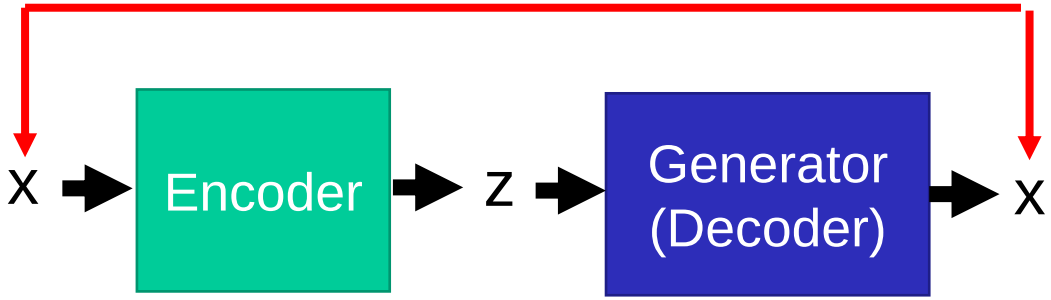
$$z^* = arg\min_z L\big(G(z), x^T\big)$$

Gradient Descent

Difference between G(z) and x^T

- Pixel-wise
- By another network

- ***Method 2***



- ***Method 3***

Using the results from ***method 2*** as the initialization of ***method 1***

Editing Photos

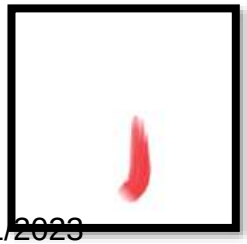
- z_0 is the code of the input image

Using discriminator to check the image is realistic or not

image

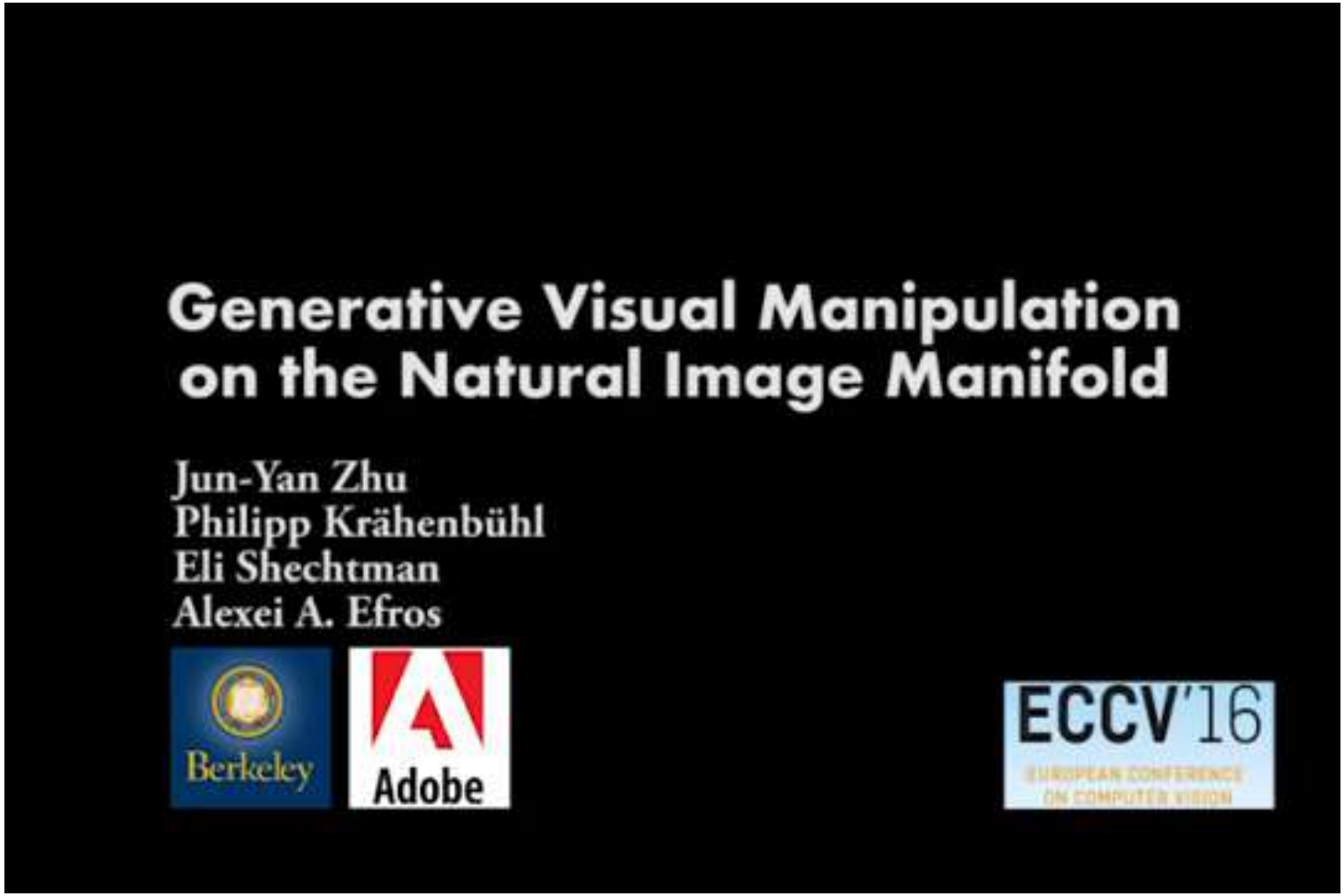
$$z^* = arg\min_z U\big(G(z)\big) + \lambda_1 \|z - z_0\|^2 - \lambda_2 D\big(G(z)\big)$$

Not too far away from the original image



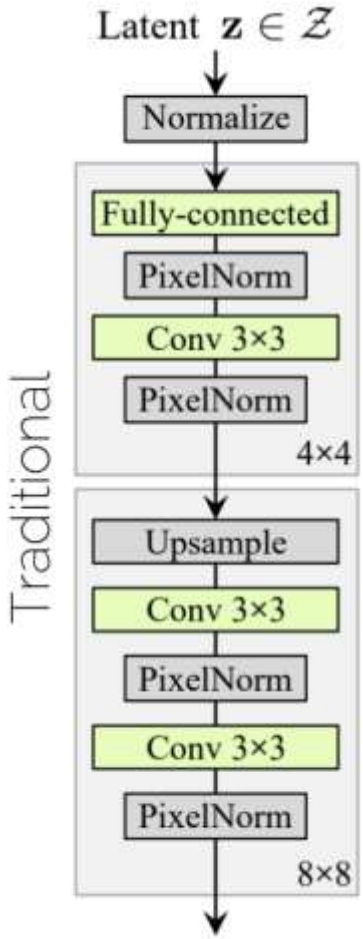
Does it fulfill the constraint of editing?

GAN + AE: attribute editing



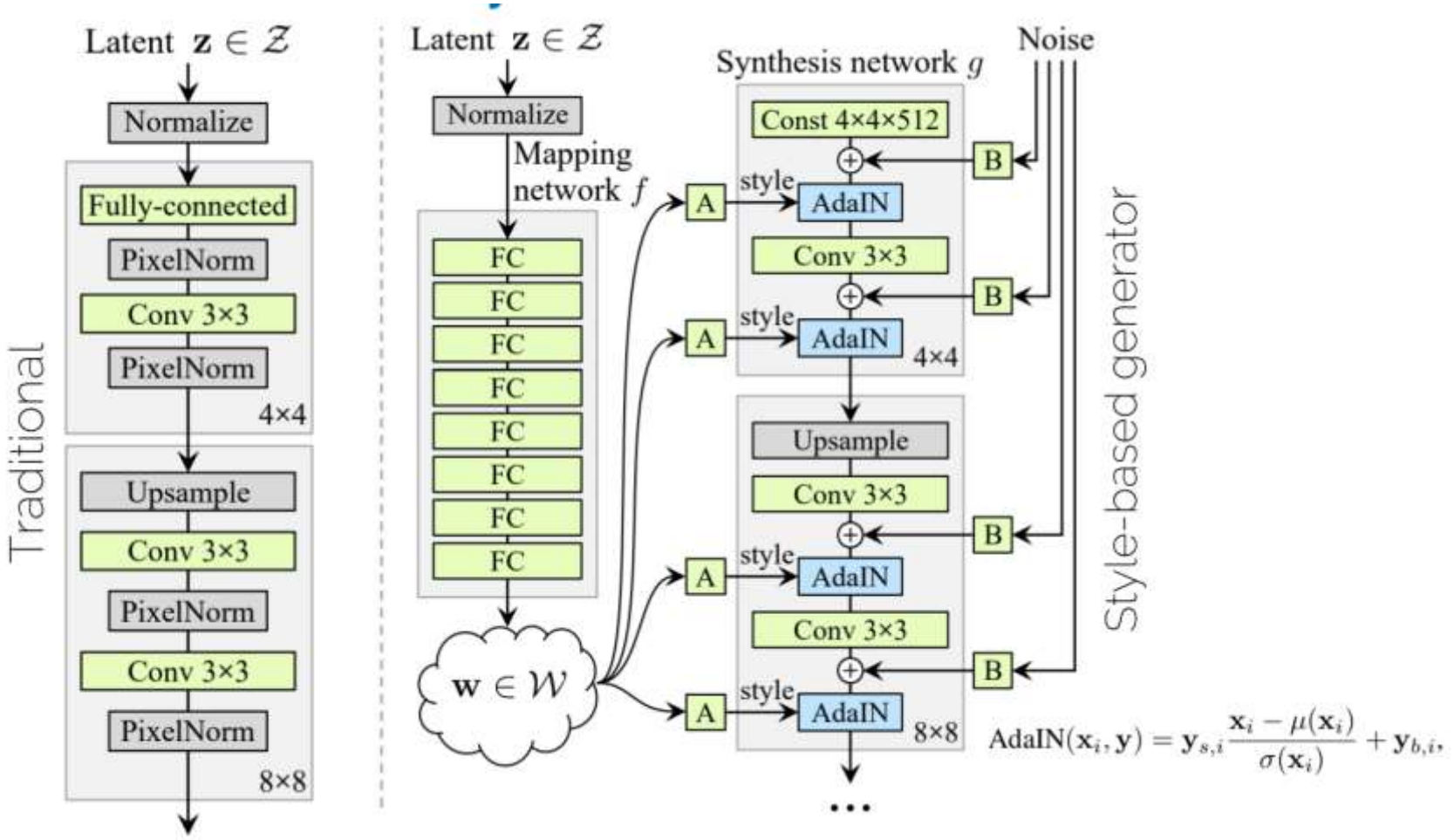
[Zhu et al. 2016] Generative Visual Manipulation on the Natural Image Manifold

StyleGAN



[Karras et al. 19]: StyleGAN

StyleGAN



[Karras et al. 19]: StyleGAN

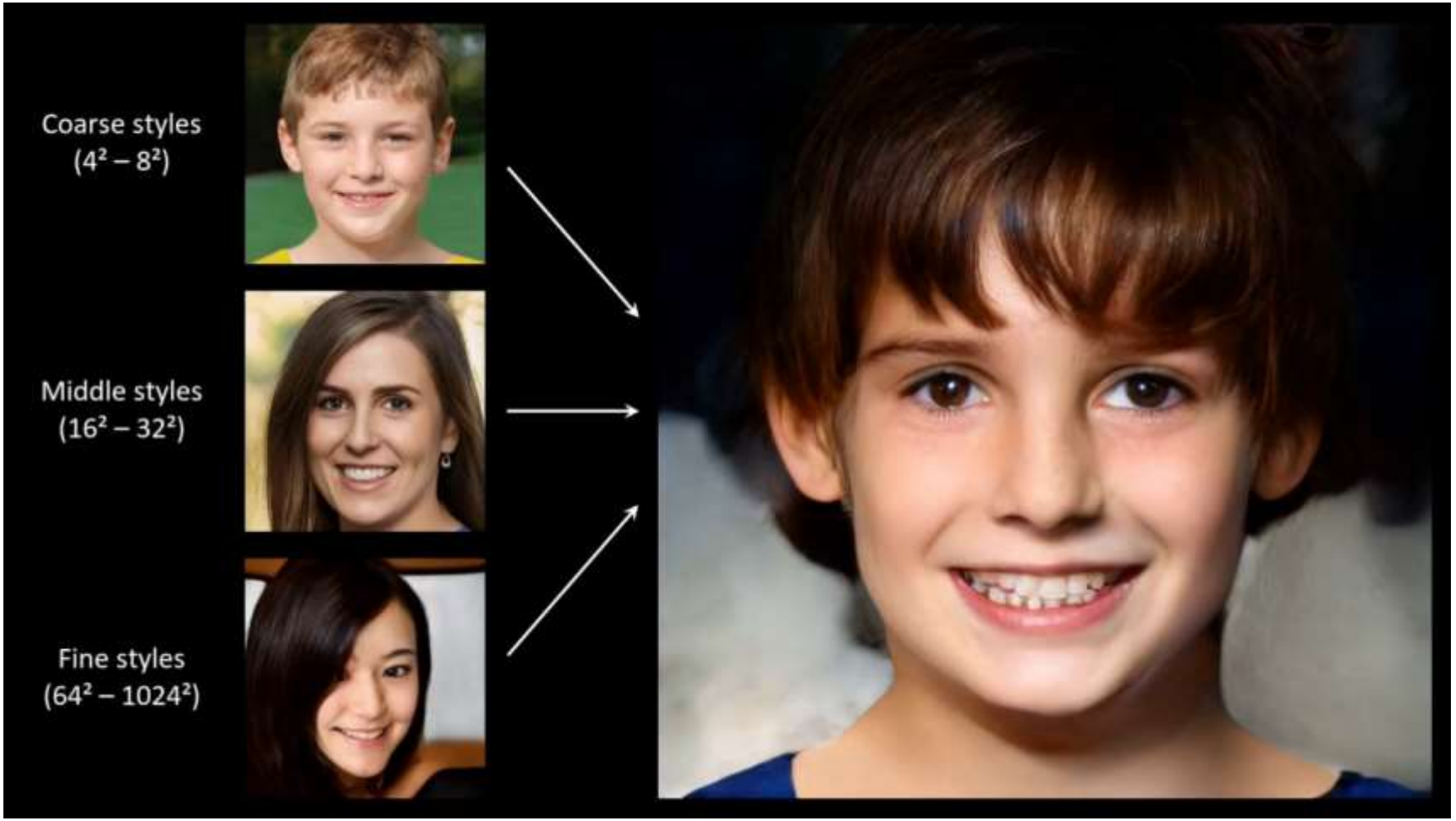
StyleGAN

- FID on 50k gen. images
- Architecture is similar to Progressive Growing GAN

	Method	CelebA-HQ	FFHQ
A	Baseline Progressive GAN [30]	7.79	8.04
B	+ Tuning (incl. bilinear up/down)	6.11	5.25
C	+ Add mapping and styles	5.34	4.85
D	+ Remove traditional input	5.07	4.88
E	+ Add noise inputs	**5.06**	4.42
F	+ Mixing regularization	5.17	**4.40**

[Karras et al. 19]: StyleGAN

StyleGAN

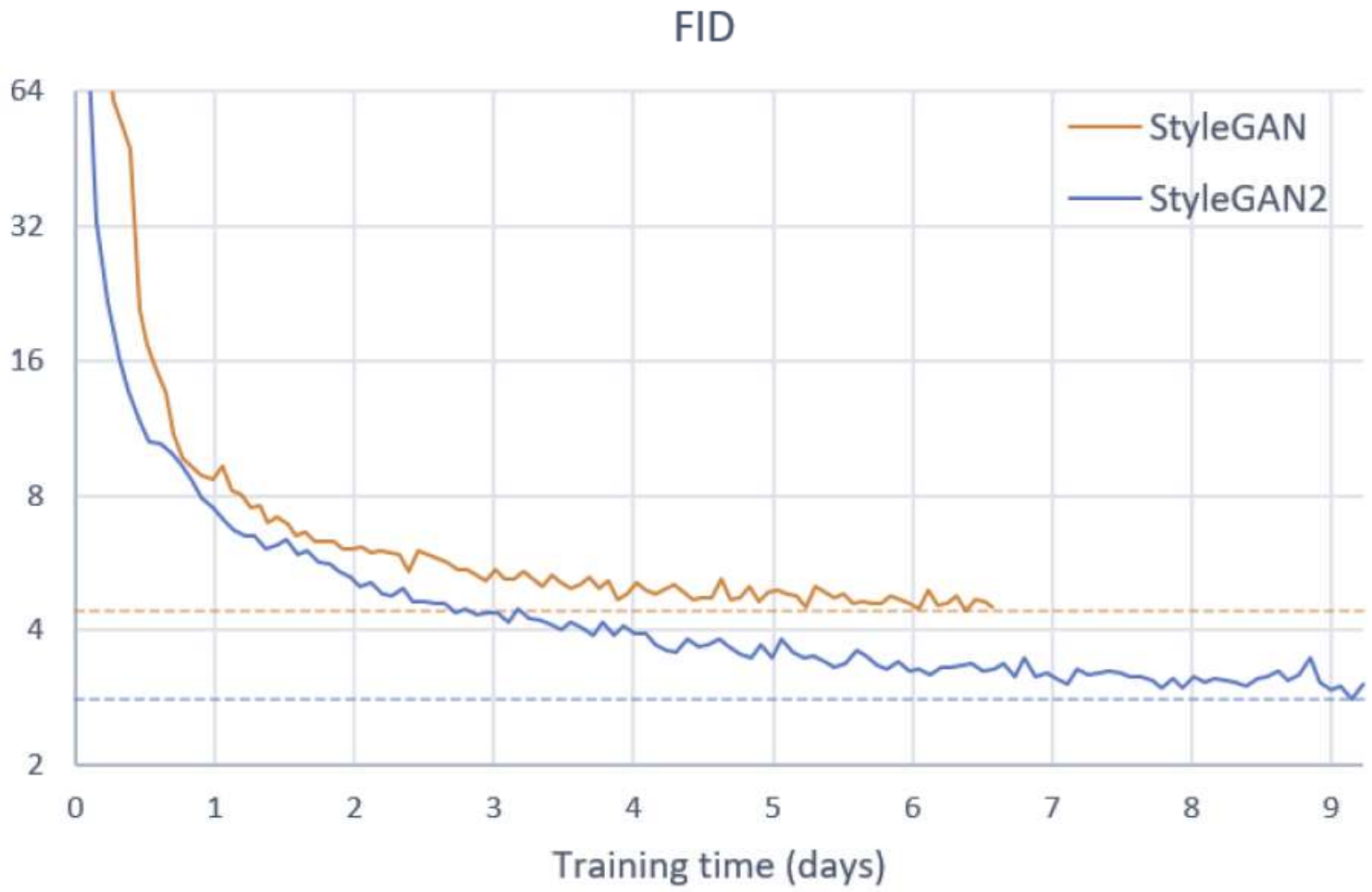


[Karras et al. 19]: StyleGAN

StyleGAN2

- Interesting analysis about design choices!
  - https://arxiv.org/pdf/1912.04958.pdf
  - https://github.com/NVlabs/stylegan2

StyleGAN v.s StyleGAN2



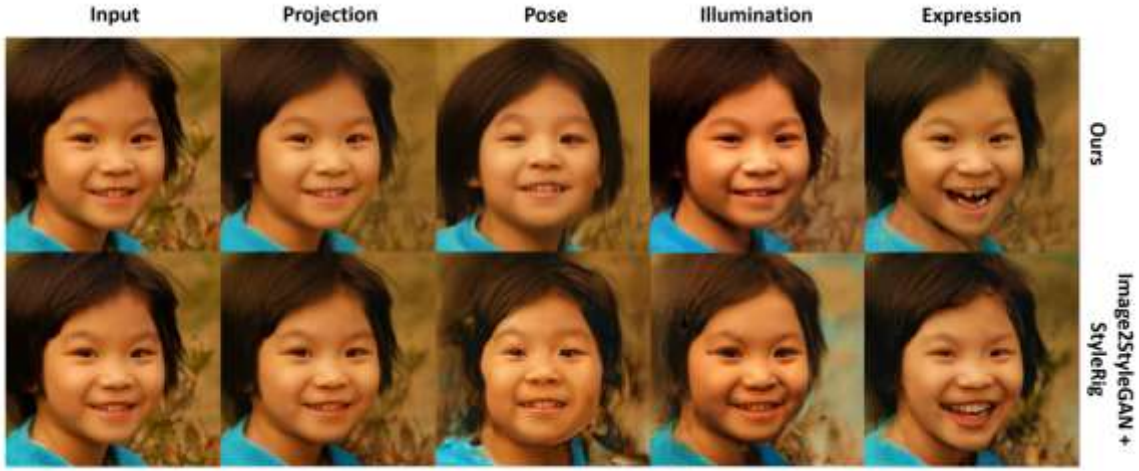
StyleGAN for style-transfer

- Keep basic and medium levels
- Fine-tune it on the fine leves

Latent space of StyleGAN

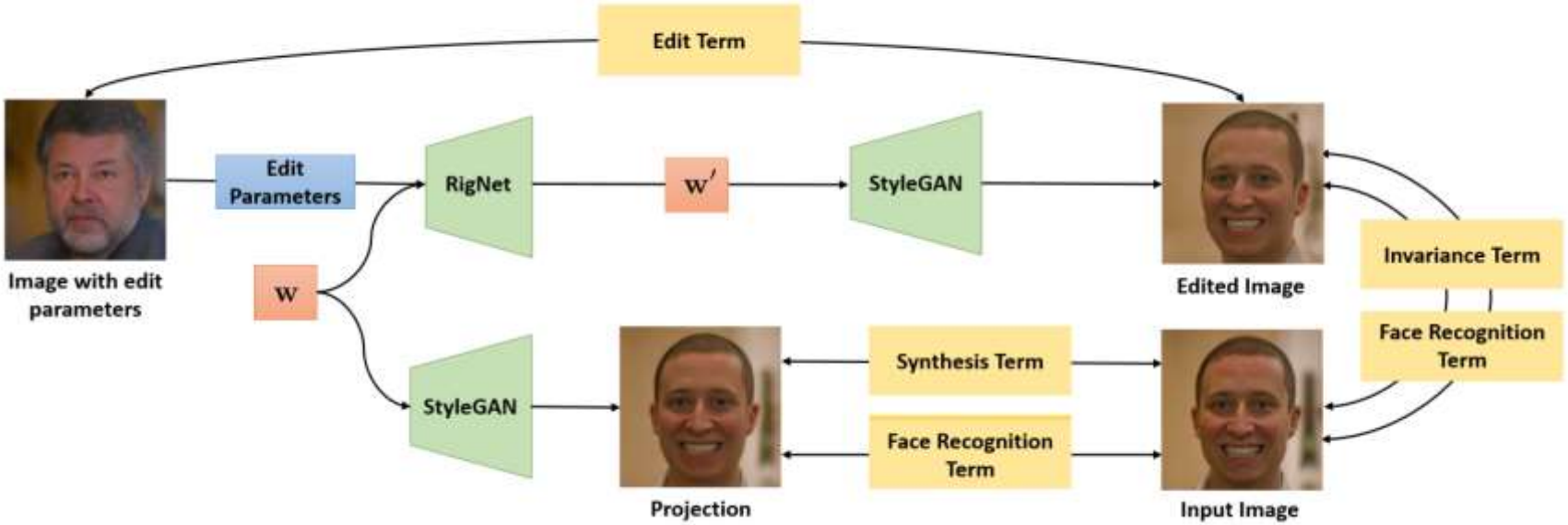
- StyleRig: Rigging StyleGAN for 3D Control over Portrait Images
  - Face rig-like control over StyleGAN

- PIE: Portrait Image Embedding for Semantic Control
  - Embedding in the latent space to edit real images



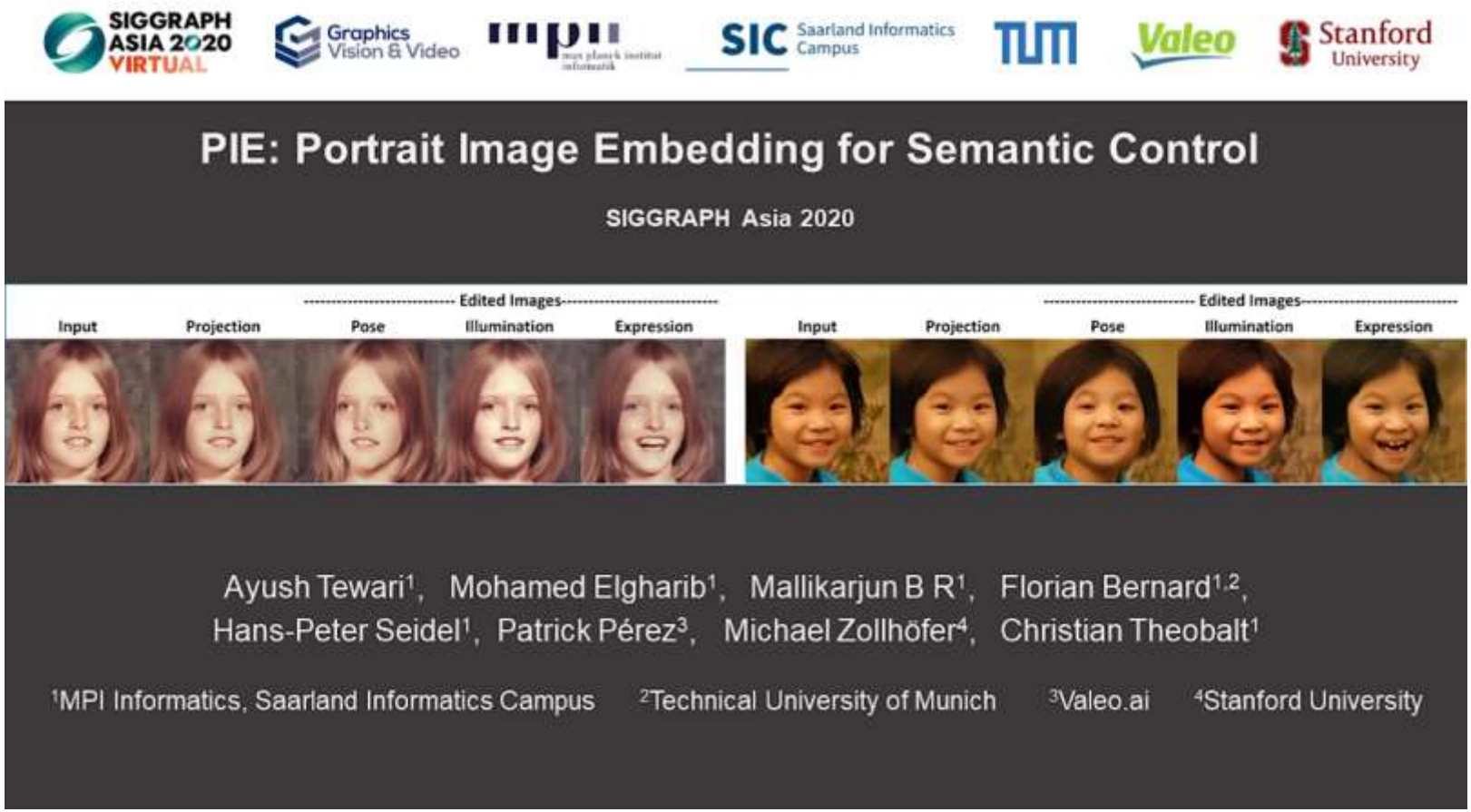
PIE: Portrait Image Embedding

- Utilize pretrained StyleGAN



[TEWARI et al. 19] PIE: Portrait Image Embedding for Semantic Control

PIE: Portrait Image Embedding

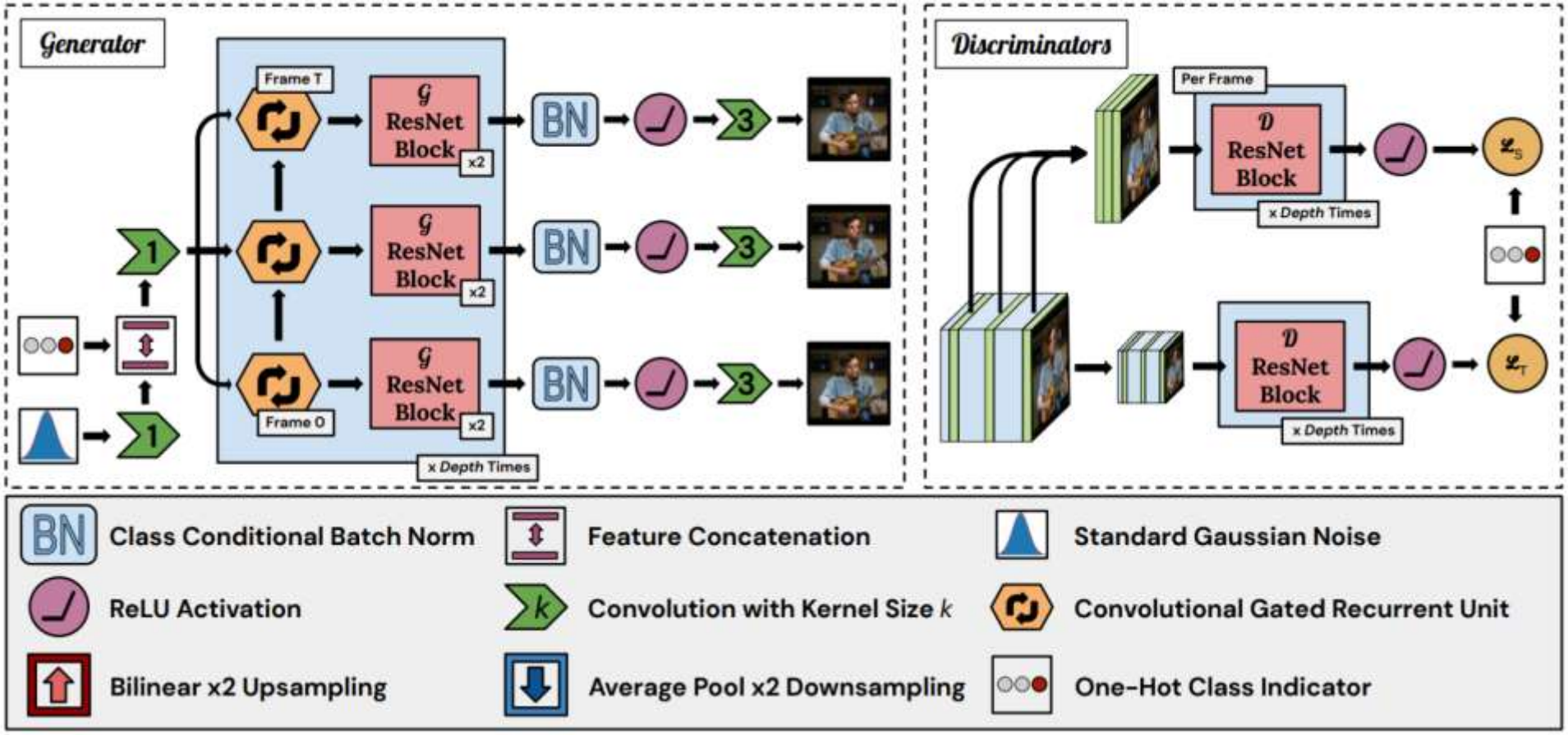


[TEWARI et al. 19] PIE: Portrait Image Embedding for Semantic Control

GANs on Videos

- Two options
  - Single random variable z seeds entire video (all frames)
    - Very high dimensional output
    - How to do for variable length?
    - Future frames deterministic given past
  - Random variable z for each frame of the video
    - Need conditioning for future from the past
    - How to get combination of past frames + random vectors during training
- General issues
  - Temporal coherency
  - Drift over time (many models collapse to mean image)

GANs on Videos: DVD-GAN

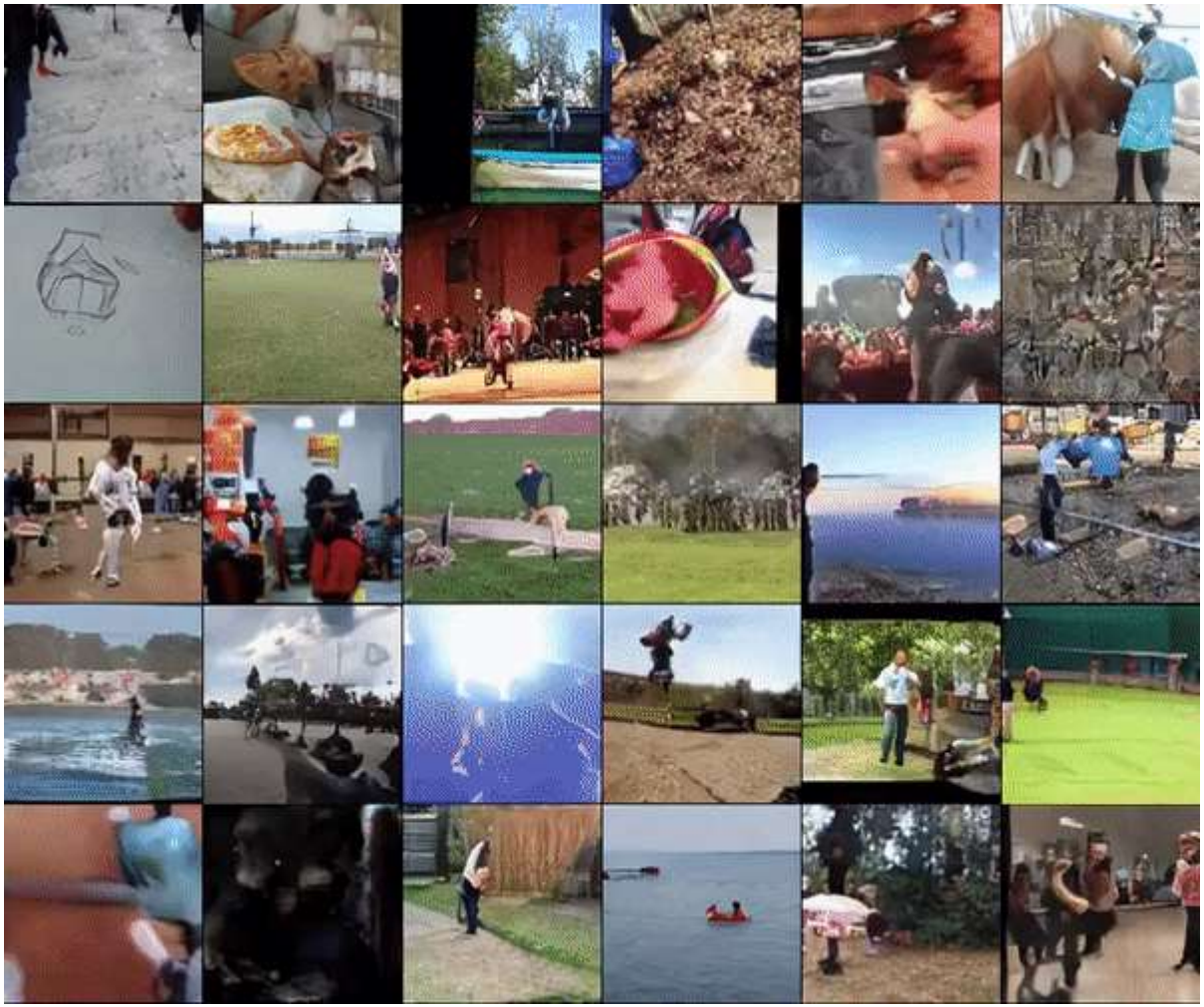


[Clark et al. 2019] Adversarial Video Generation on Complex Datasets

GANs on Videos: DVD-GAN

[Clark et al. 2019] Adversarial Video Generation on Complex Datasets

GANs on Videos: DVD-GAN



[Clark et al. 2019] Adversarial Video Generation on Complex Datasets

GANs on Videos: DVD-GAN

- Trained on Kinetics-600 dataset
  - 256 x 256, 128 x 128, and 64 x 64
  - Lengths of up 48 frames
- This is state of the art!
- Videos from scratch still incredibly challenging

Conditional GANs on Videos

- Challenge:
  - Each frame is high quality, but temporally inconsistent

Video-to-Video Synthesis

- Sequential Generator:

$$p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T) = \prod_{t=1}^{T} p(\tilde{\mathbf{x}}_t|\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t}).$$

past L generated frames past L source frames

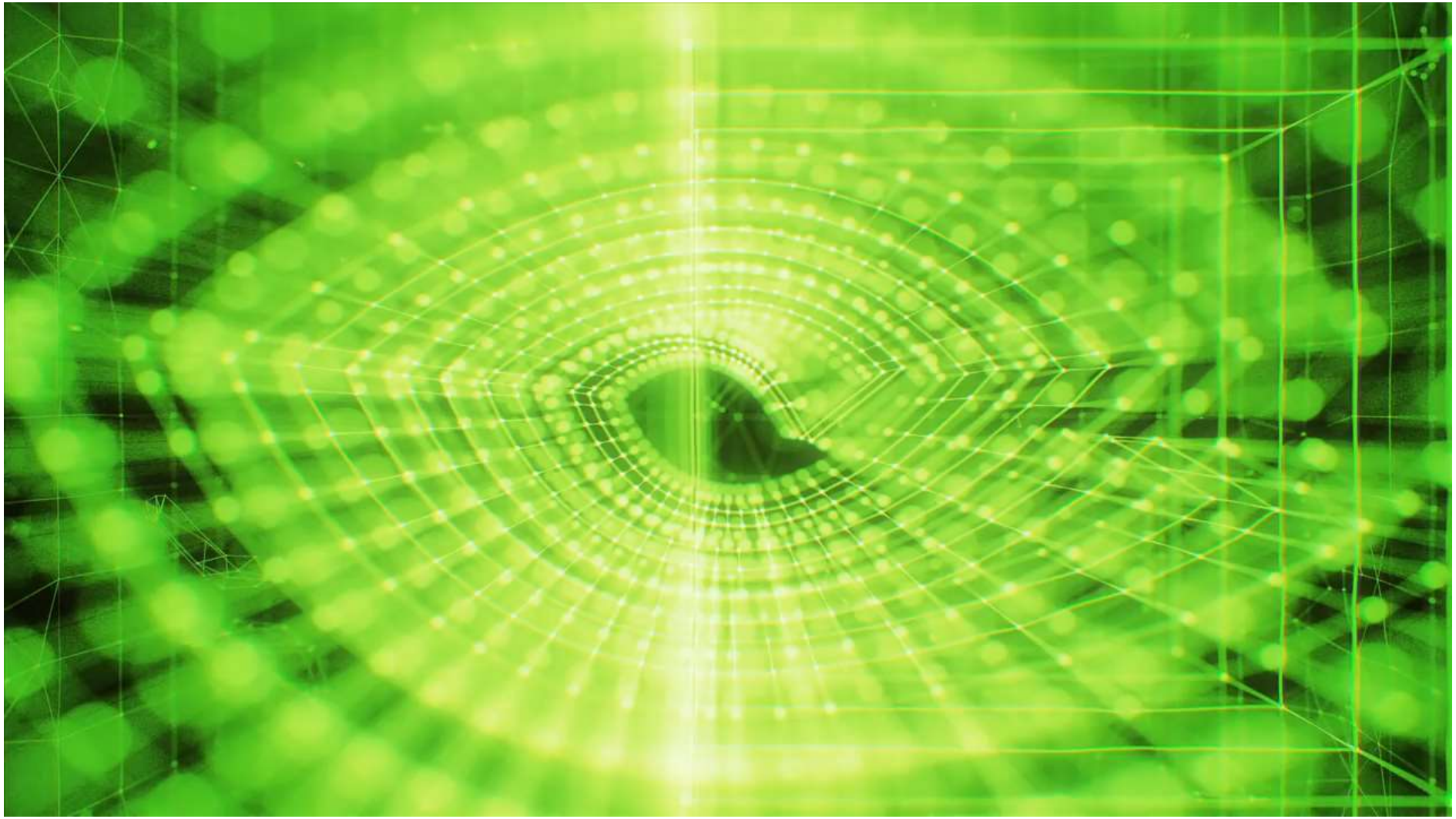
(Set L = 2)

- Conditional Image Discriminator D_i (is it real image)
- Conditional Video Discriminator D_v (temp. consistency via flow)
- Full Learning Objective:

$$\min_F \left(\max_{D_I} \mathcal{L}_I(F, D_I) + \max_{D_V} \mathcal{L}_V(F, D_V) \right) + \lambda_W \mathcal{L}_W(F)$$

[Wang et al. 2018] Vid2Vid

Video-to-Video Synthesis



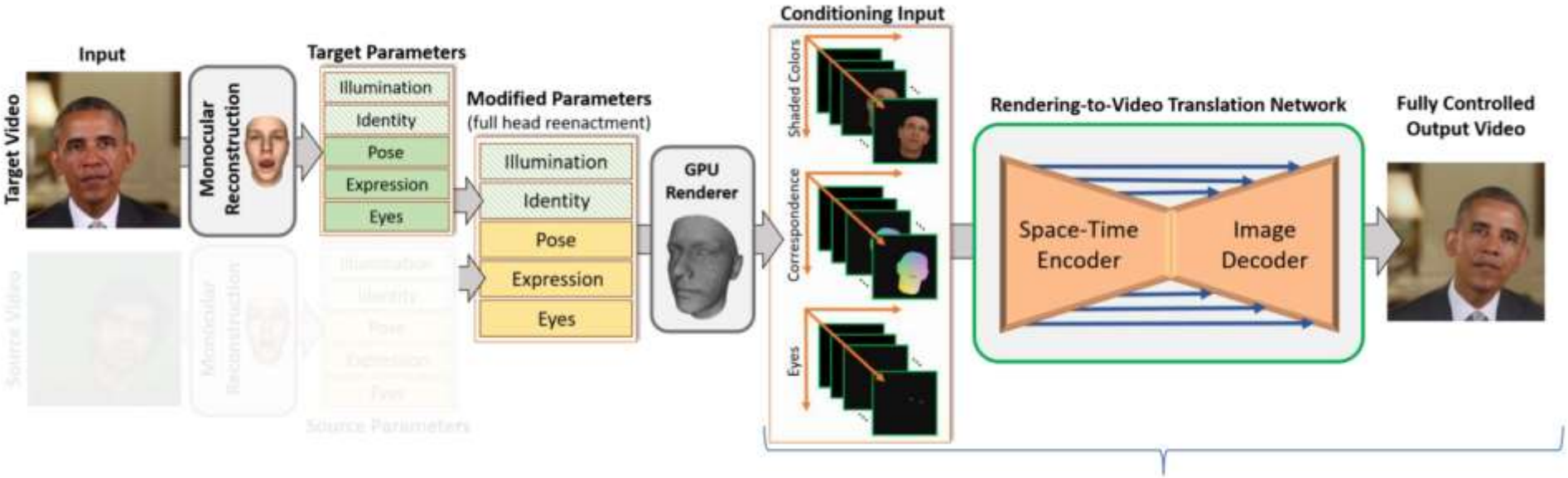
[Wang et al. 2018] Vid2Vid

Video-to-Video Synthesis

Key ideas:

- Separate discriminator for temporal parts
  - In this case based on optical flow
- Consider recent history of previous frames
- Train all of it jointly

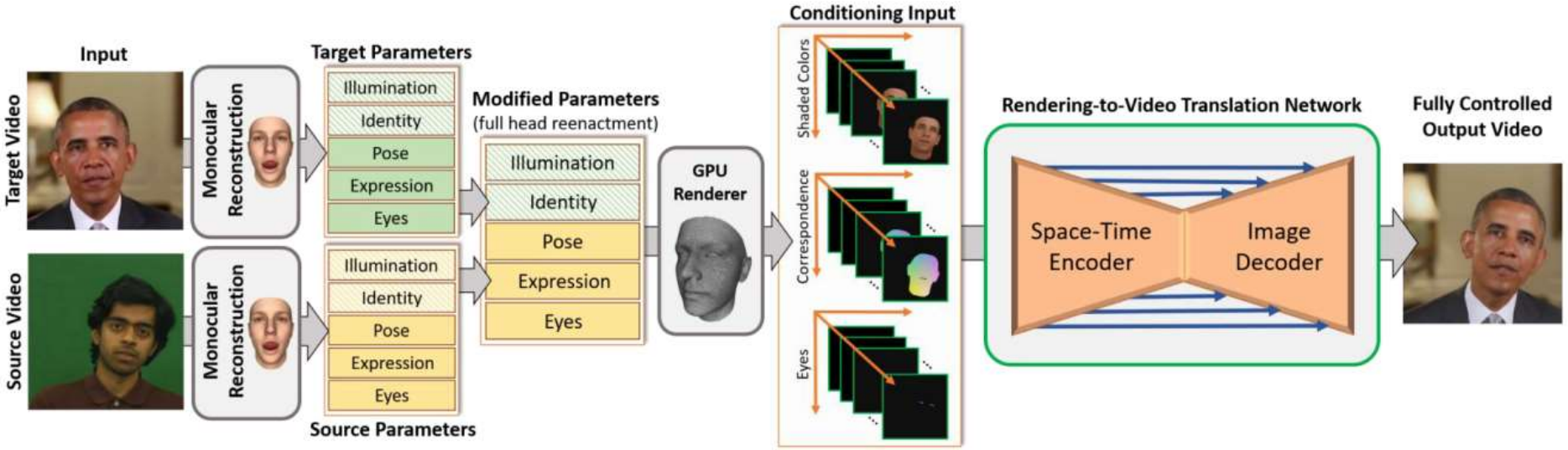
Deep Video Portraits



Similar to “Image-to-Image Translation” (Pix2Pix) [Isola et al.]

Siggraph’18 [Kim et al 2018]: Deep Portraits

Deep Video Portraits



Siggraph'18 [Kim et al 2018]: Deep Portraits

Deep Video Portraits

Source Sequence

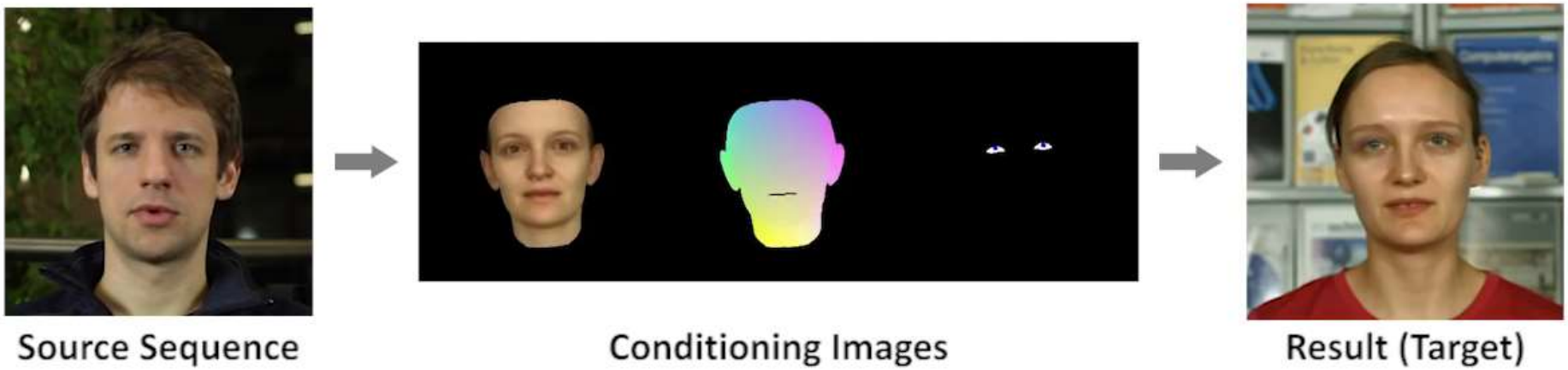
Conditioning Images

Result (Target)

Neural Network converts synthetic data to realistic video

Siggraph'18 [Kim et al 2018]: Deep Portraits

Deep Video Portraits



Siggraph'18 [Kim et al 2018]: Deep Portraits

Deep Video Portraits

Siggraph'18 [Kim et al 2018]: Deep Portraits

Deep Video Portraits



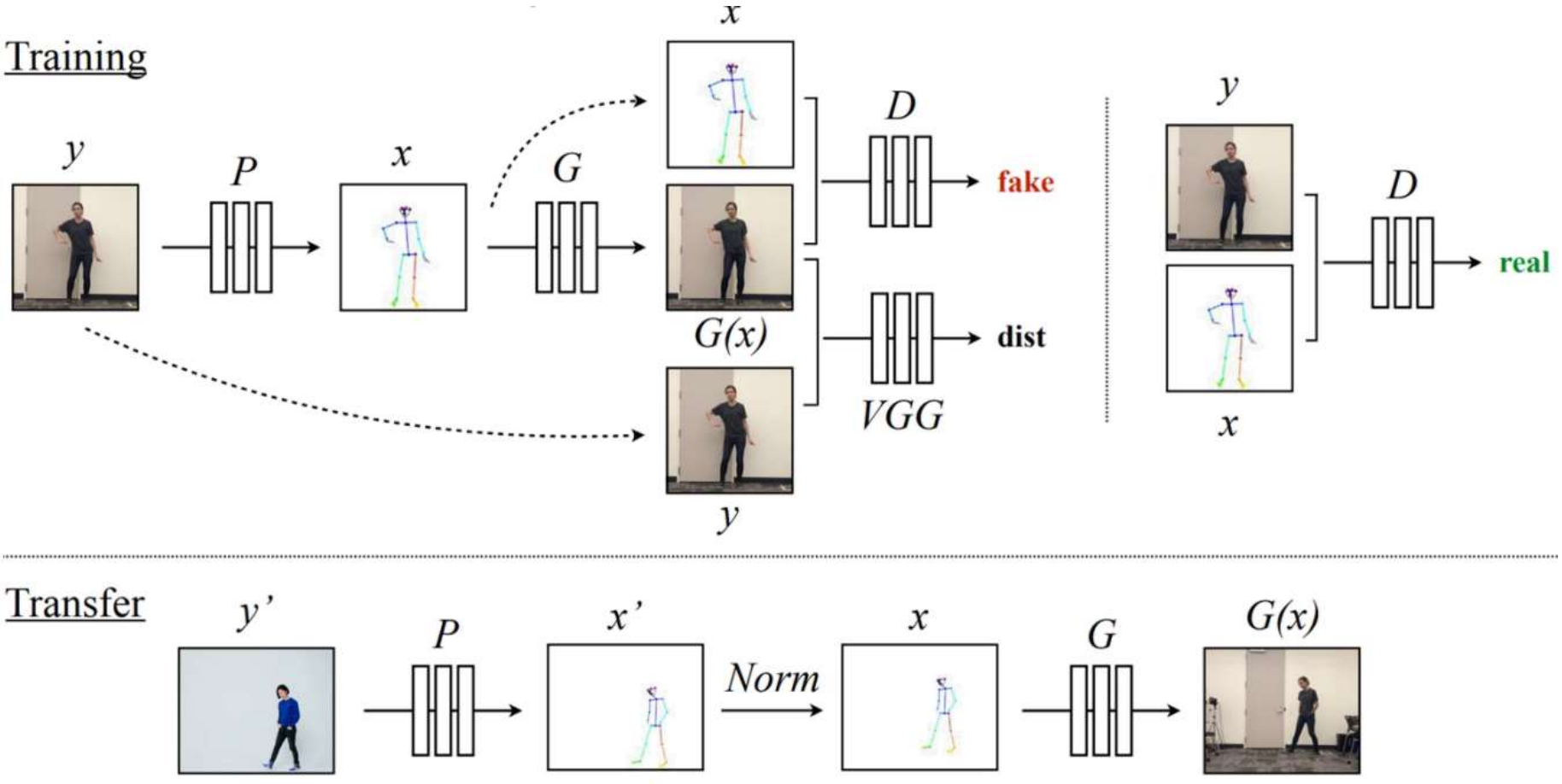
Siggraph'18 [Kim et al 2018]: Deep Portraits

Deep Video Portraits

Key ideas:

- Synthetic data for tracking is great anchor / stabilizer
- Overfitting on small datasets works pretty well
- Need to stay within training set w.r.t. motions
- No real learning; essentially, optimizing the problem with SGD

Everybody Dance Now



[Chan et al. '18] Everybody Dance Now

Everybody Dance Now

[Chan et al. ’18] Everybody Dance Now

Everybody Dance Now

- cGANs work with different input
- Requires consistent input i.e., accurate tracking
- Network has no explicit 3D notion

[Chan et al. ’18] Everybody Dance Now

Everybody Dance Now

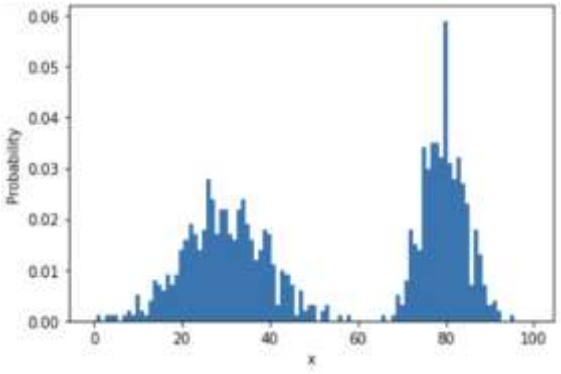
Key ideas:

- Conditioning via tracking seems promising!
- Tracking quality translates to resulting image quality
- Tracking human skeletons is less developed than faces
  - Temporally it's not stable… (e.g., OpenPose etc.)
- Fun fact, there were like 4 papers with a similar idea that appeared around the same time… (even more papers recently)

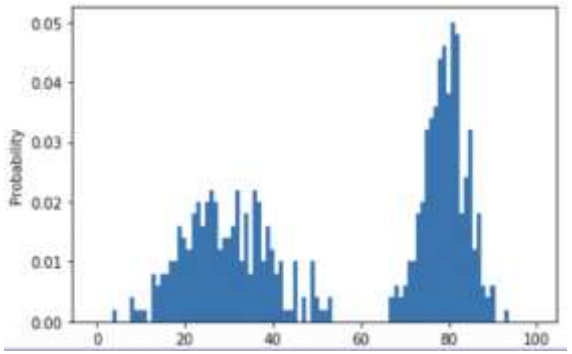
Recap of generative modeling so far

- Assume that all data comes from a data distribution that is unknown to us.
- The goal of deep generative modeling is to train neural networks that can **learn** this distribution from its samples (i.e., a fixed size training dataset)
- We generate new data by **sampling** from the learned distribution
- In practice, models are trained to maximize **expected log-likelihood** or minimize **divergence** between the learned and empirical data distributions

Training distribution



Samples from learned distribution



Prior methods

- VAE

$$
\begin{aligned}
L_{\text{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\
\theta^*, \phi^* &= \arg\min_{\theta, \phi} L_{\text{VAE}}
\end{aligned}
$$

$$
\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}
$$

- GAN

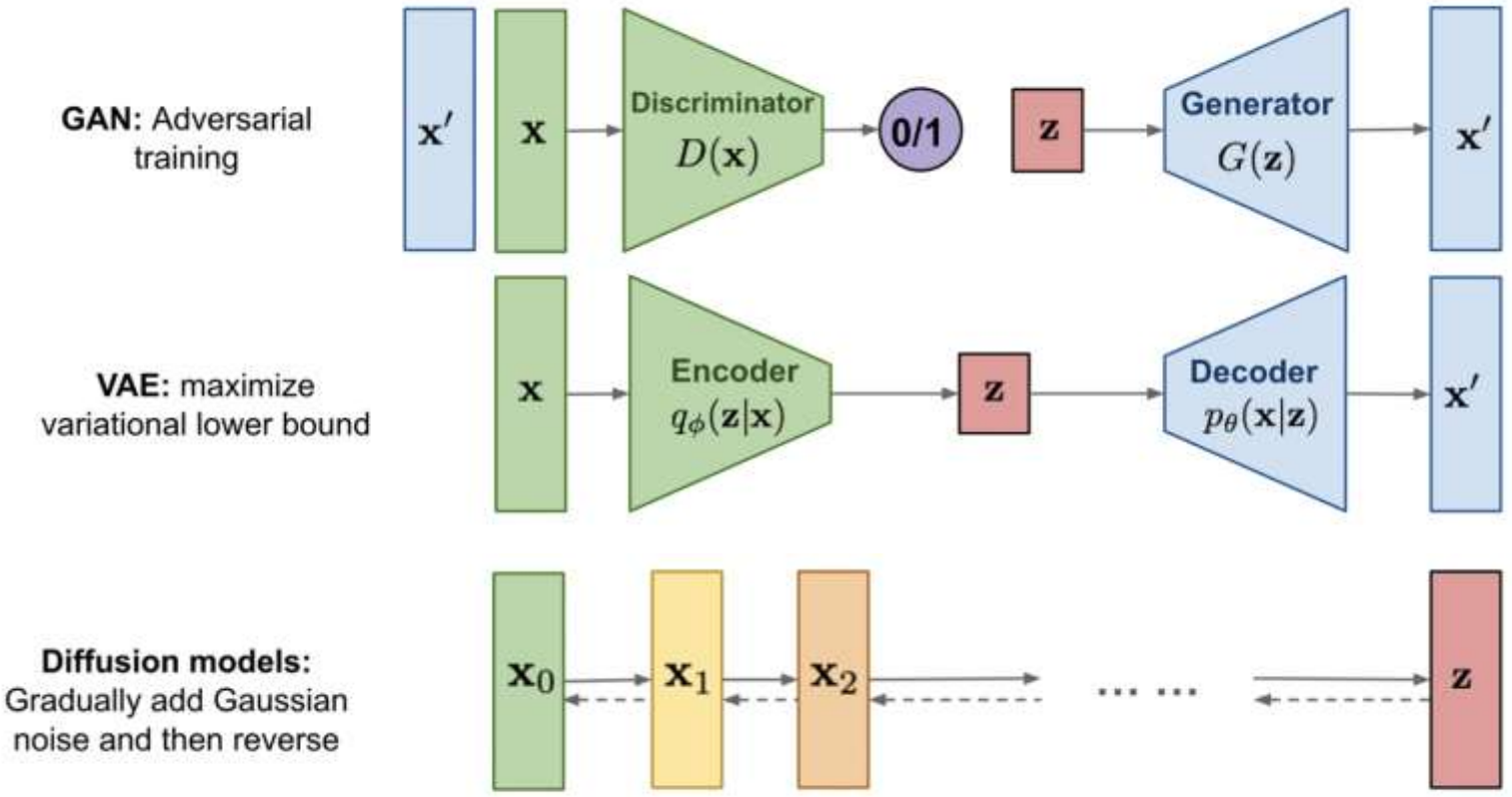
$$
\begin{aligned}
\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\
&= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]
\end{aligned}
$$

$$
\boxed{L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2\log 2}
$$

The Landscape of Generative Models

Sampling from noise

- A quick paradigm ……



Diffusion models

- What if we add a bunch of Gaussian noise to an image?

Diffusion models

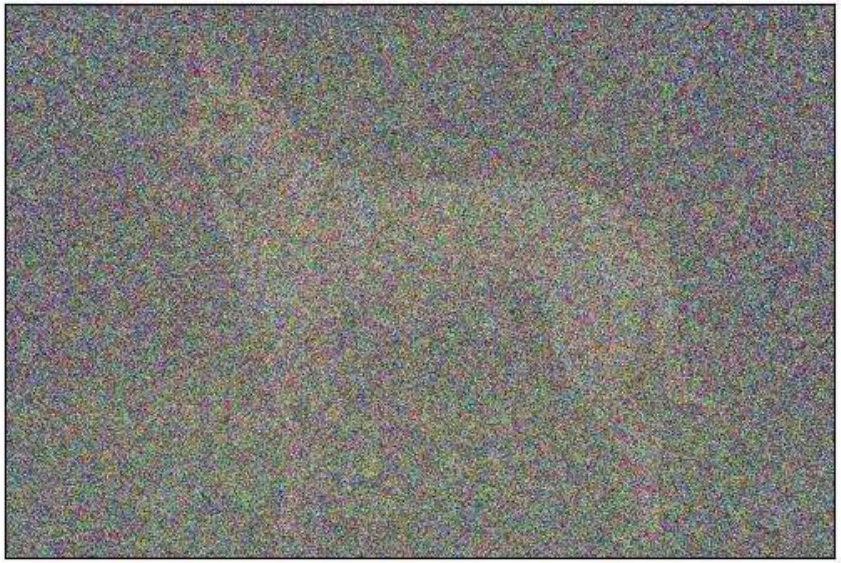
- and again…

Diffusion models

- and again…

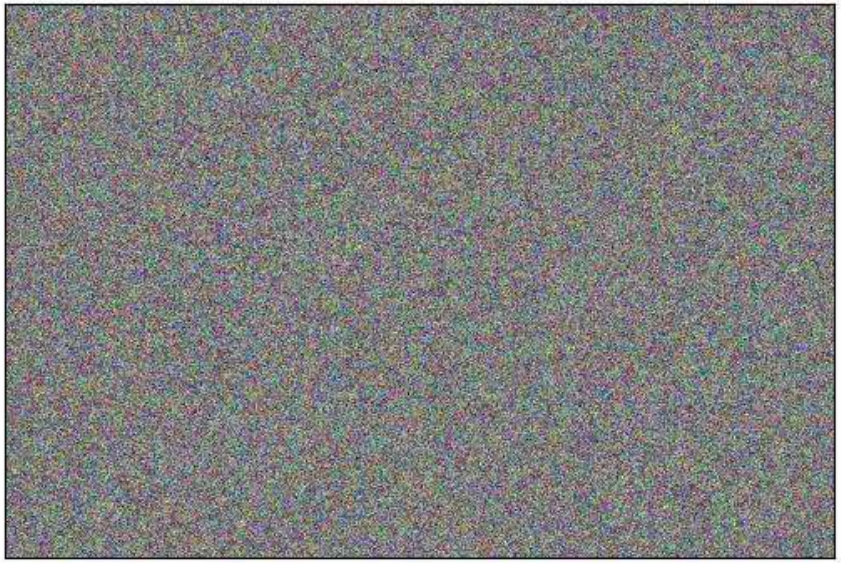
Diffusion models

- and again…



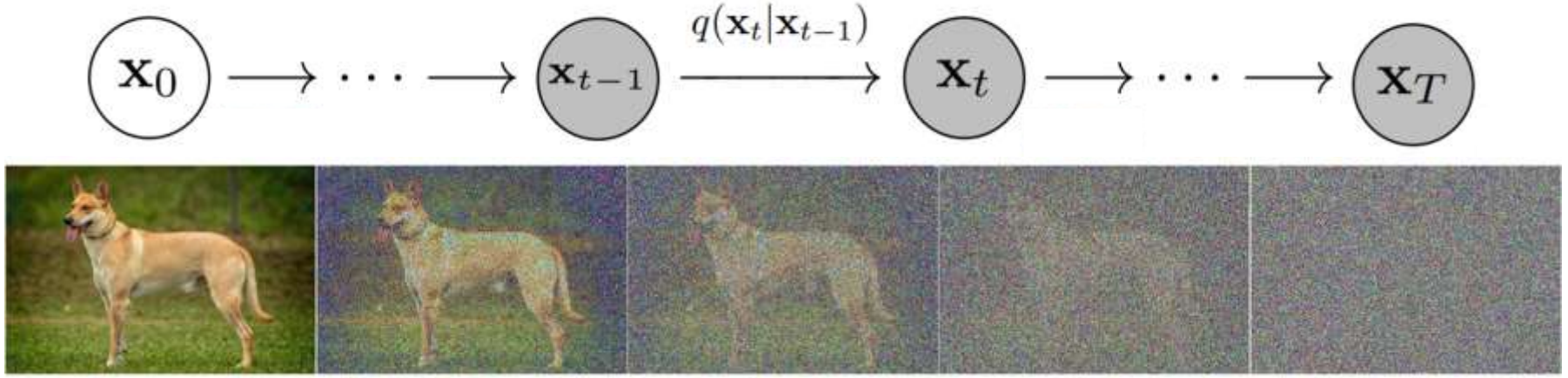
Diffusion models

- … until it resembles pure noise



Diffusion models

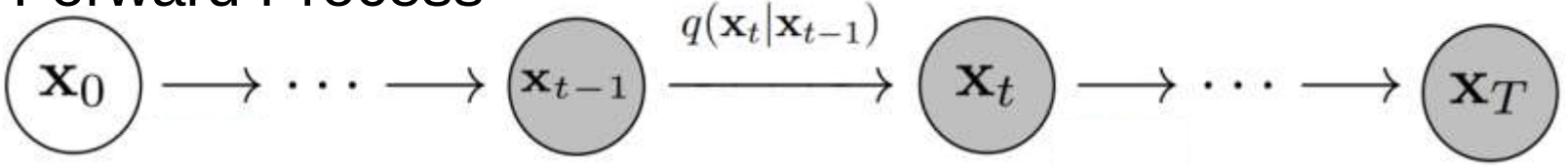
- Forward Process



- Take a datapoint **x_0** and gradually add very small amounts of Gaussian noise to it
- Let **x_t** be the datapoint after t iterations
- This is called the **forward diffusion process**
- Repeat this process for **T** steps — over time, more and more features of the original input are destroyed until you get something resembling **pure noise**

Diffusion models

- Forward Process



- More formally, we update each image over time as

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I)$$

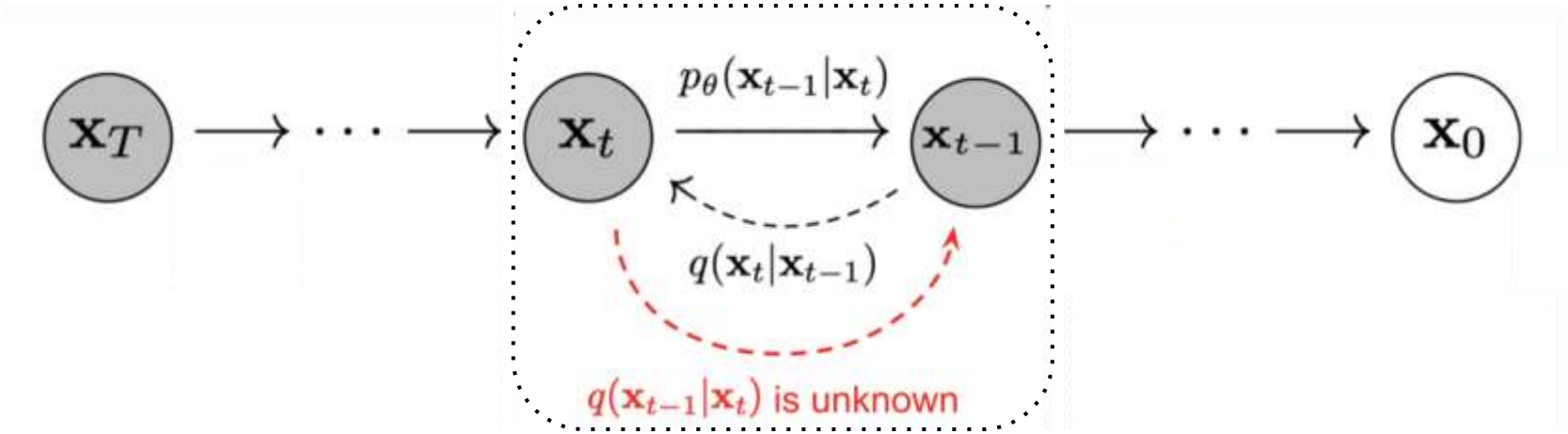
where $\{\beta_t \in (0,1)\}_{t=1}^{T}$

is called the **noise schedule** (basically a hyperparameter describing how much noise to add at a given timestep).
The update above can equivalently be written as a sampling process from the following Gaussian distribution:

$$q(x_t \mid x_{t-1}) \sim \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Diffusion models

- Can we go in the other direction?



Diffusion models

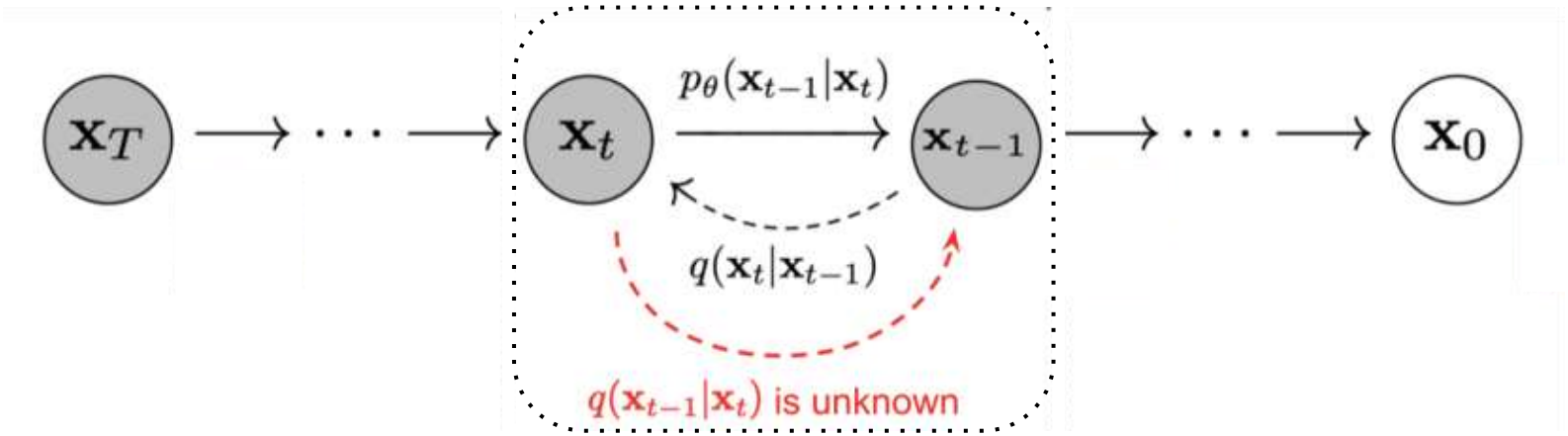
- Reverse Process

- The goal of a diffusion model is to **learn the reverse denoising process** to iteratively **undo** the forward process
- In this way, the reverse process appears as if it is generating new data from random noise!

Diffusion models

- Reverse Process

We are given **q(x_t | x_{t-1})**. How do we find **q(x_{t-1} | x_t)**?



Diffusion models

- Finding the exact distribution is hard
- Remember Bayes rule?

$$f(\theta \mid x) = \frac{f(\theta, x)}{f(x)} = \frac{f(\theta)\, f(x \mid \theta)}{f(x)} \quad \Rightarrow \quad q(x_{t-1} \mid x_t) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$$

$$q(x_t) = \int q(x_t \mid x_{t-1}) q(x_{t-1}) \mathrm{d}x$$

- This is computationally intractable
  - Need to integrate over the whole data distribution to find **q(x_t)** and **q(x_{t-1})**
  - Where else have we seen this dilemma?
- We still need the posterior distribution to carry out the reverse process. Can we approximate this somehow?

Diffusion models

- What does the final reverse process look like?
- In practice, we choose **our noise schedule** such that the forward process steps are **very small**. Thus, we approximate the reverse posterior distributions as Gaussians and **learn** their parameters (i.e., the mean and variance) via neural networks

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

- Cool, we now have an idea of what the model looks like. How do we train it?

Diffusion models

- A preliminary objective

We want to maximize the log-likelihood of the data generated by a reverse process.

Remember that VAEs tried to do something similar but they maximized a lower bound on the likelihood instead because the actual likelihood is computationally intractable

$$-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})$$

We can apply the same trick to diffusion!

$$-L = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] \leq \log p_\theta(\mathbf{x}_0)$$

Diffusion models

- A simplified objective

Recall the forward process:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I)$$

Skipping the derivation here, but you can show that

$$L = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[w_t \left\| \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(\mathbf{x}_t, t)\right\|_2^2\right]$$

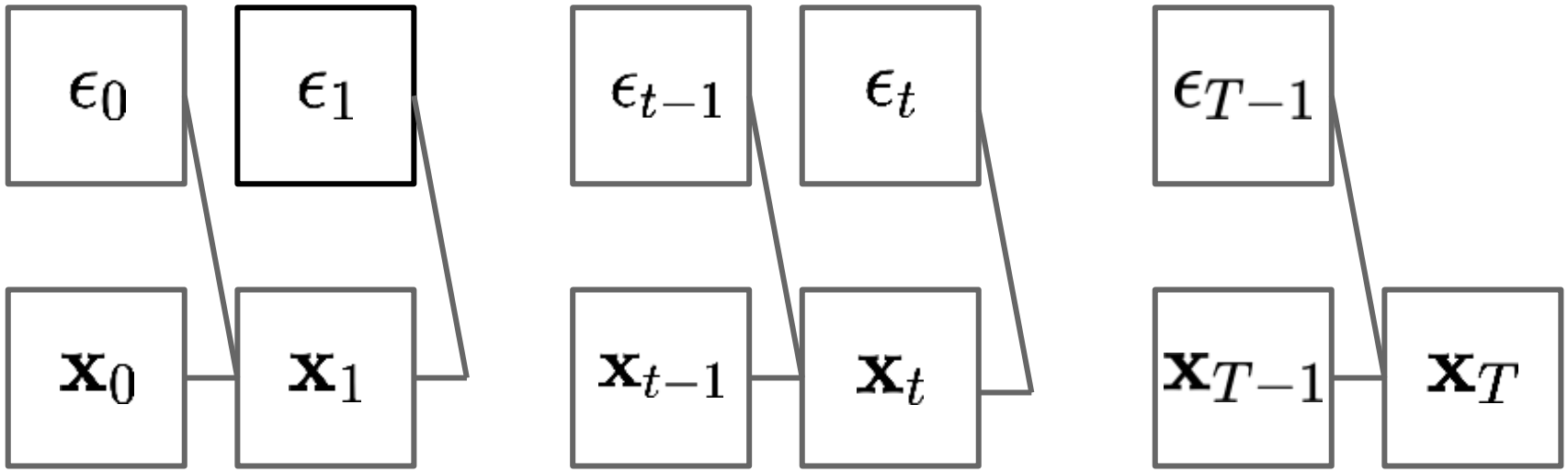
our estimate for x_{t-1}

where

$$\alpha_t = 1 - \beta_t \quad \text{and} \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_t$$

Diffusion models

- A simplified objective
- Forward process re-visualized



Diffusion models

- An even more simplified objective!

Instead of predicting an estimate for **x_{t-1}**, it's actually better to predict the noise ϵ that was added to **x_{t-1}** to produce **x_t**.

In other words,

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I)$$

Don't predict this

Predict this instead

Thus, our final loss becomes

$$L = \mathbb{E}_{\mathbf{x}_0,\, \epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t,\, t)\|_2^2\right]$$

$$= \mathbb{E}_{\mathbf{x}_0,\, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\, t\right)\right\|_2^2\right]$$

Diffusion models

- Training

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Diffusion models

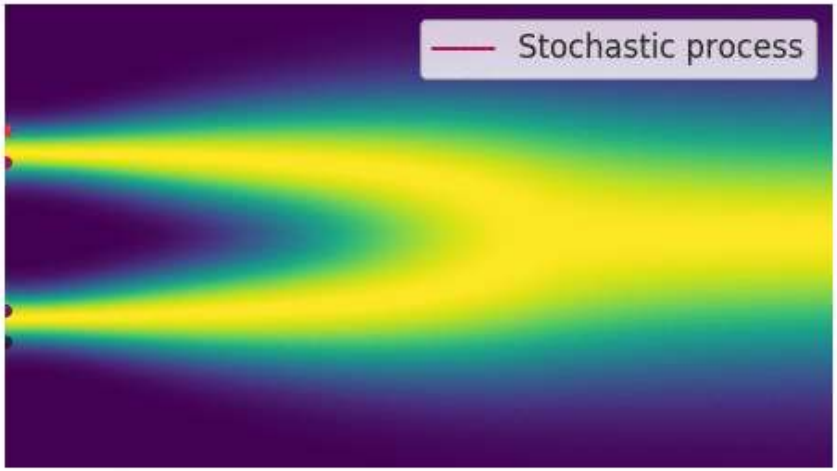
- Sampling

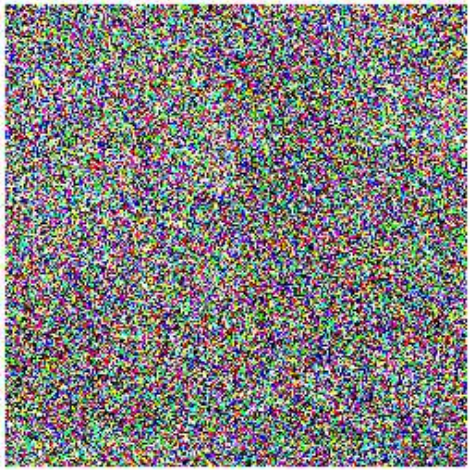
Algorithm 2 Sampling

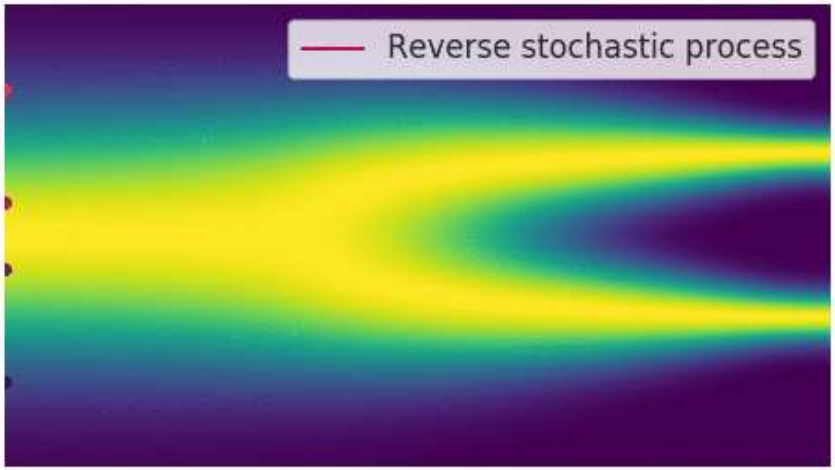
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta (\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Diffusion models

- Forward/Reverse process for Image Generation







Summary

- Variants of GANs
  - CycleGAN: Image-to-image translation with unpaired data
  - GANs on videos
- Diffusion, very quick start
- Keep working on the projects!