Multimodal Generative AI: from GANs to Diffusion

Vicky Kalogeiton

About me

- *Assistant Professor*, 2020 –
  - VISTA Group, Ecole Polytechnique, France
  - Main CV and GenAI researcher at Polytechnique

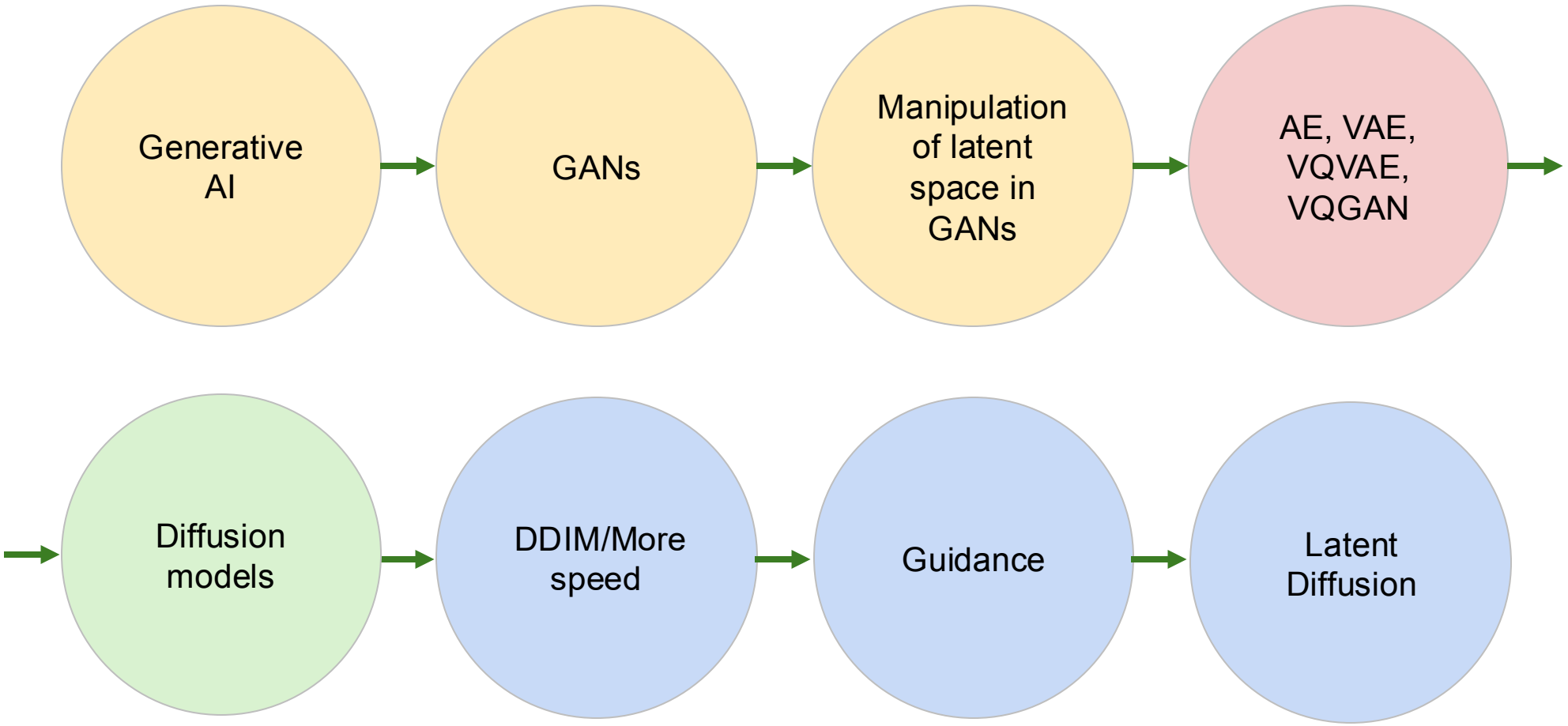
- *Research Fellow,* 2019 – 2021
- *Post-doc,* 2018 – 2020
  - Visual Geometry Group, University of Oxford, UK
  - Andrew Zisserman

VGG
UNIVERSITY OF OXFORD

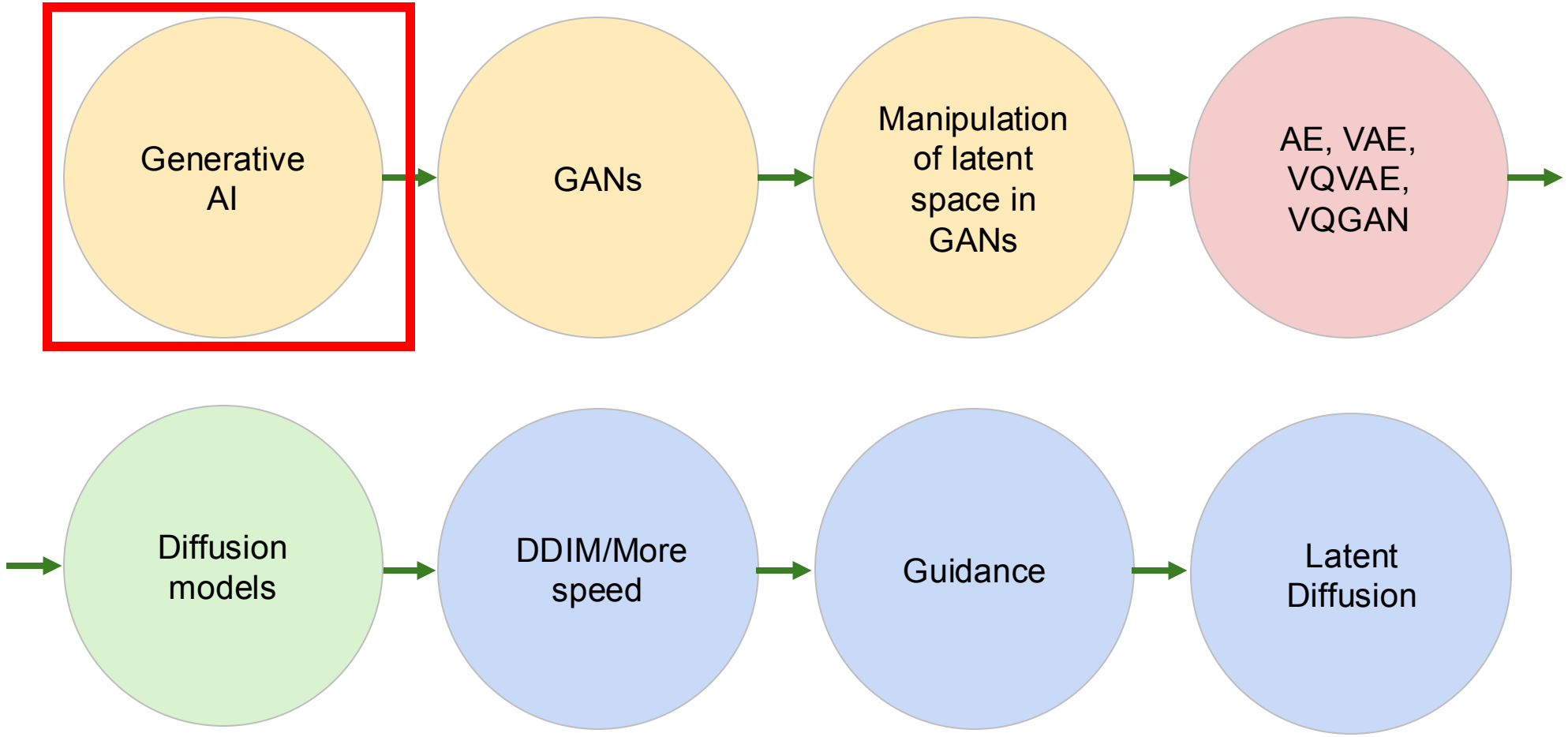
UNIVERSITY OF OXFORD

- *PhD*, 2013 – 2017
  - University of Edinburgh, UK, INRIA, Grenoble, France
  - Vittorio Ferrari, Cordelia Schmid

Multimodal Generative AI with Diffusion



Multimodal Generative AI with Diffusion



How do we understand the world?

- Our brains are like powerful computers, constantly predicting what will happen next based on past experiences

- Just like we use past knowledge to guess future events, generative AI attempts to forecast and create based on what it has learned

The Future of AI with Generative Models

- Generative AI could be a critical part of future AI advancements
- It combines creating new things and understanding causality—how one action leads to another
- Like us, these AI systems can simulate different scenarios and outcomes, helping to make better decisions

Precision Meets Creativity: A new era in AI

- Predict
- Classify
- Recommend

- Imagine
- Augment
- Create

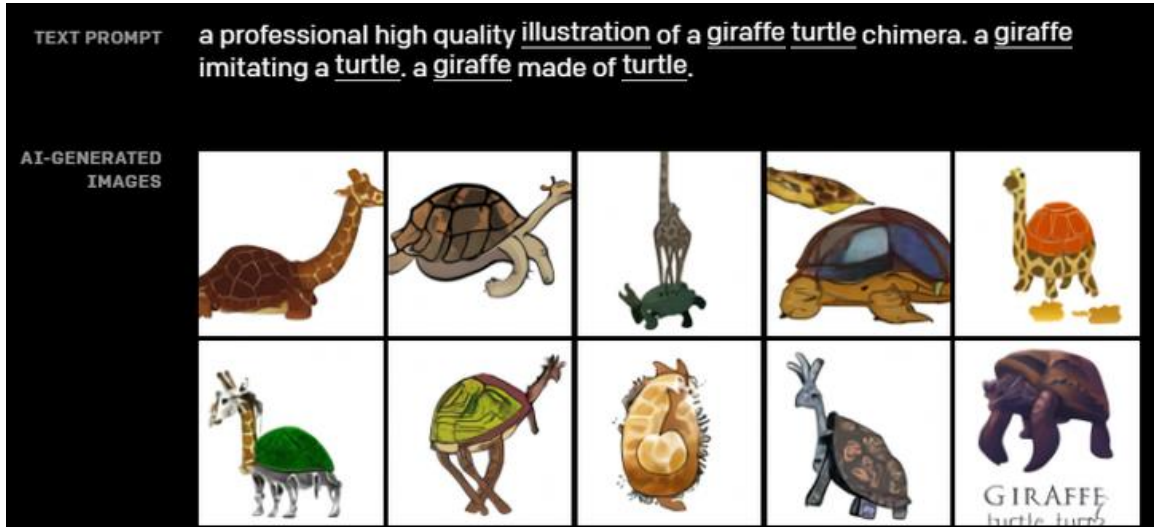
AGI? Artificial General Intelligence

Creative palette

- Images
- Audio
- Text
- 3D models
- ….

Information

Introduction



DALLE-1

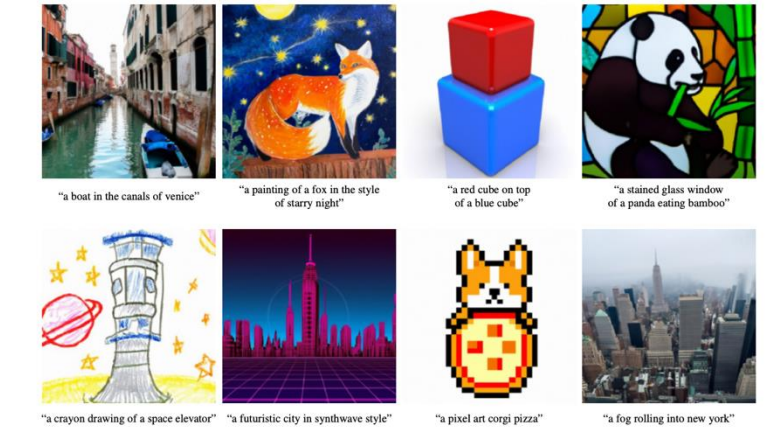
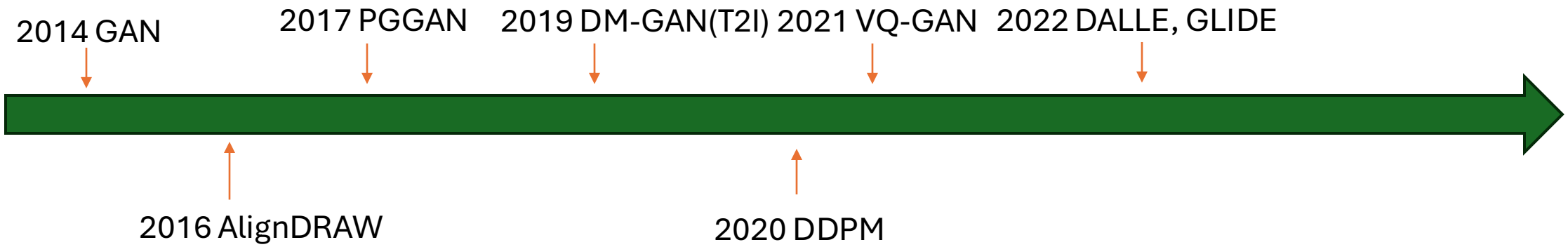
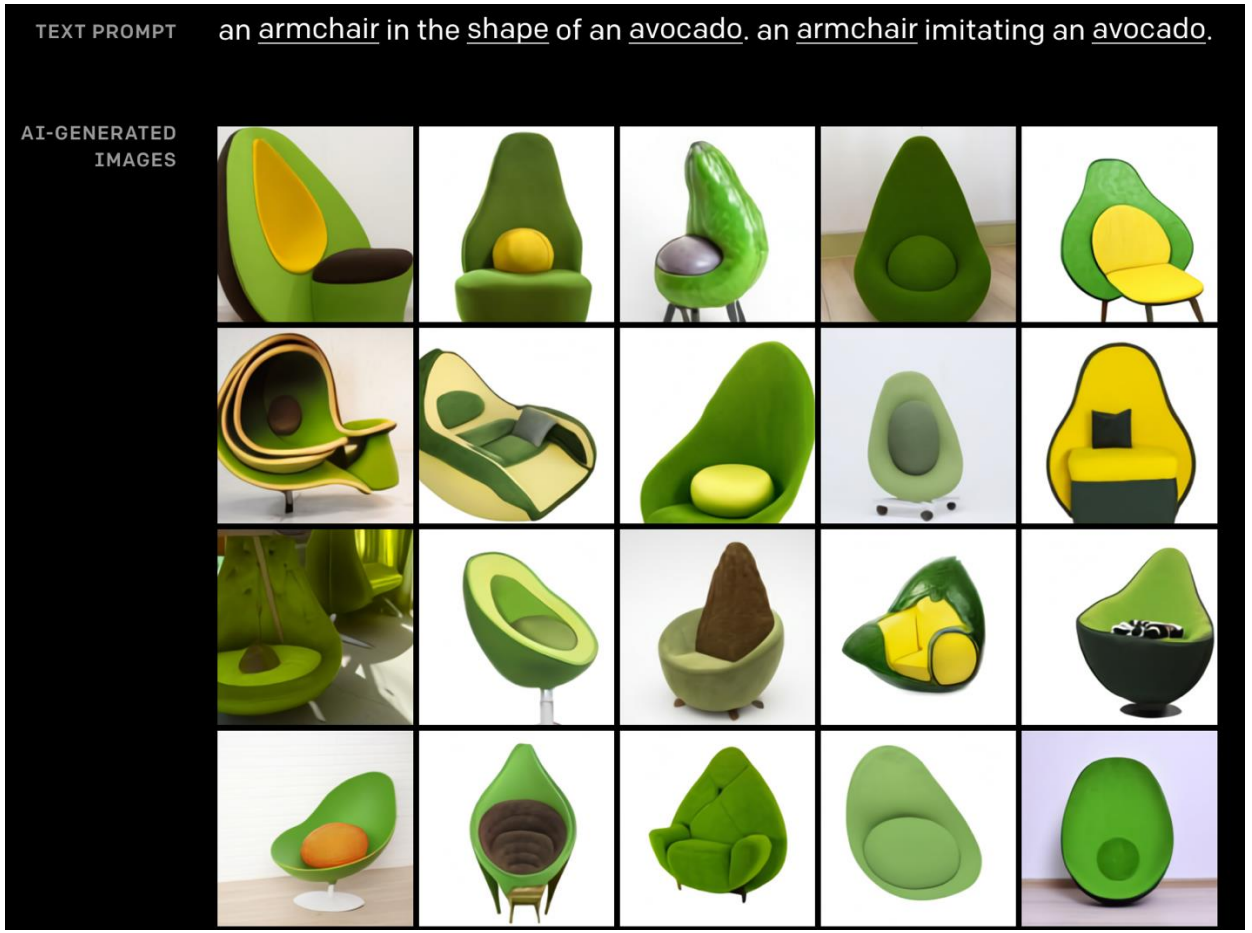


Figure 1. Selected samples from GLIDE using classifier-free guidance. We observe that our model can produce photorealistic images with shadows and reflections, can compose multiple concepts in the correct way, and can produce artistic renderings of novel concepts. For random sample grids, see Figure 17 and 18.

GLIDE



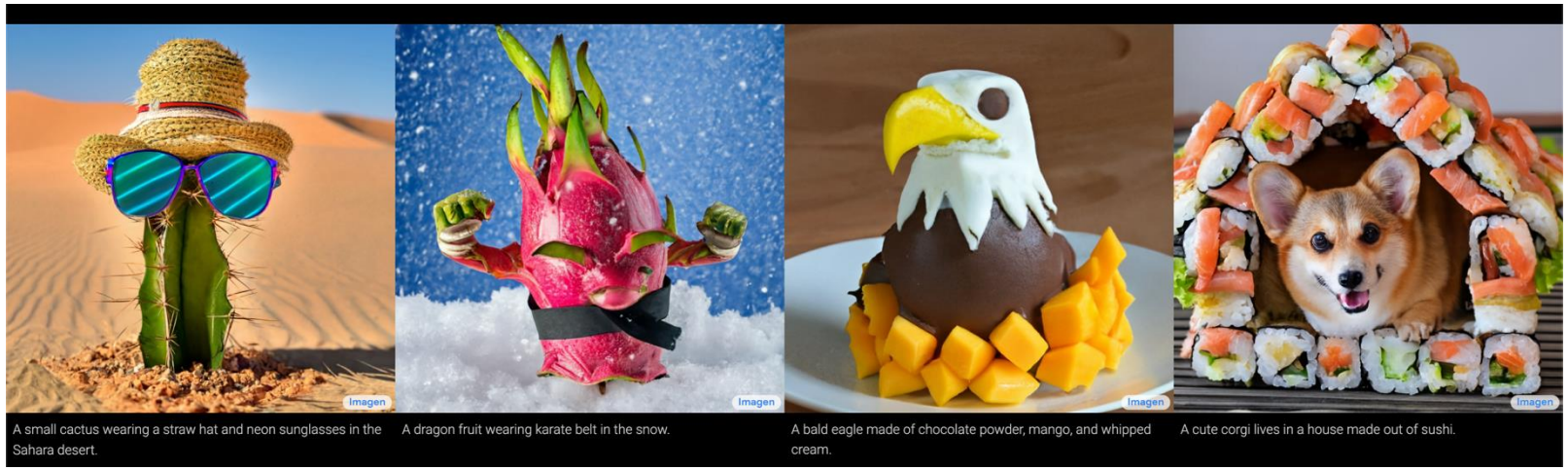
DALLE



https://openai.com/blog/dall-e/

Imagen by Google AI

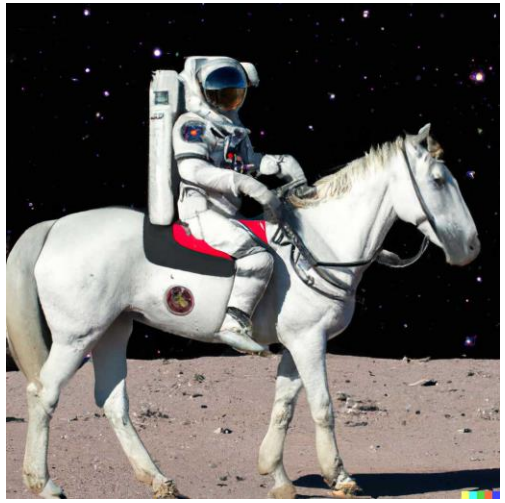
A photo of a Corgi dog riding a bike in Times Square. It is wearing sunglasses and a beach hat.



https://imagen.research.google/

Dalle-2 (Text-to-Image)

A bowl of soup as a planet in the universe



An astronaut riding a horse in a photorealistic style

Teddy bears mixing sparkling chemicals as mad scientists

Diffusion Models

OpenAI: DALL-E3

Midjourney

music, audio, animation, video, physical etc....

Stable Diffusion 3

February 2024

Make-A-Video (Text-to-Video)

A confused grizzly bear in a calculus class

A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

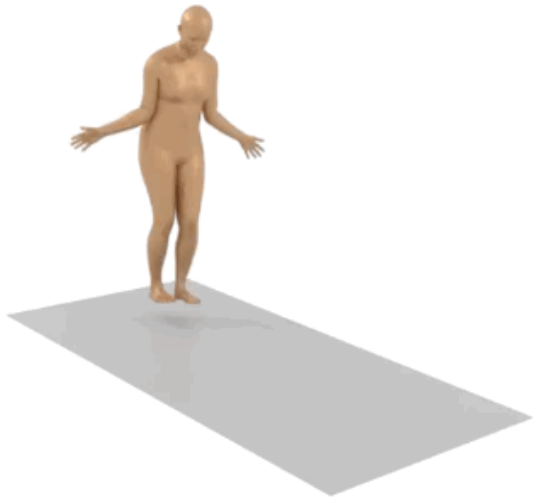
A panda playing on a swing set

SORA (Text-to-Video)

February 2024

Human Motion Diffusion (Text-to-Motion)

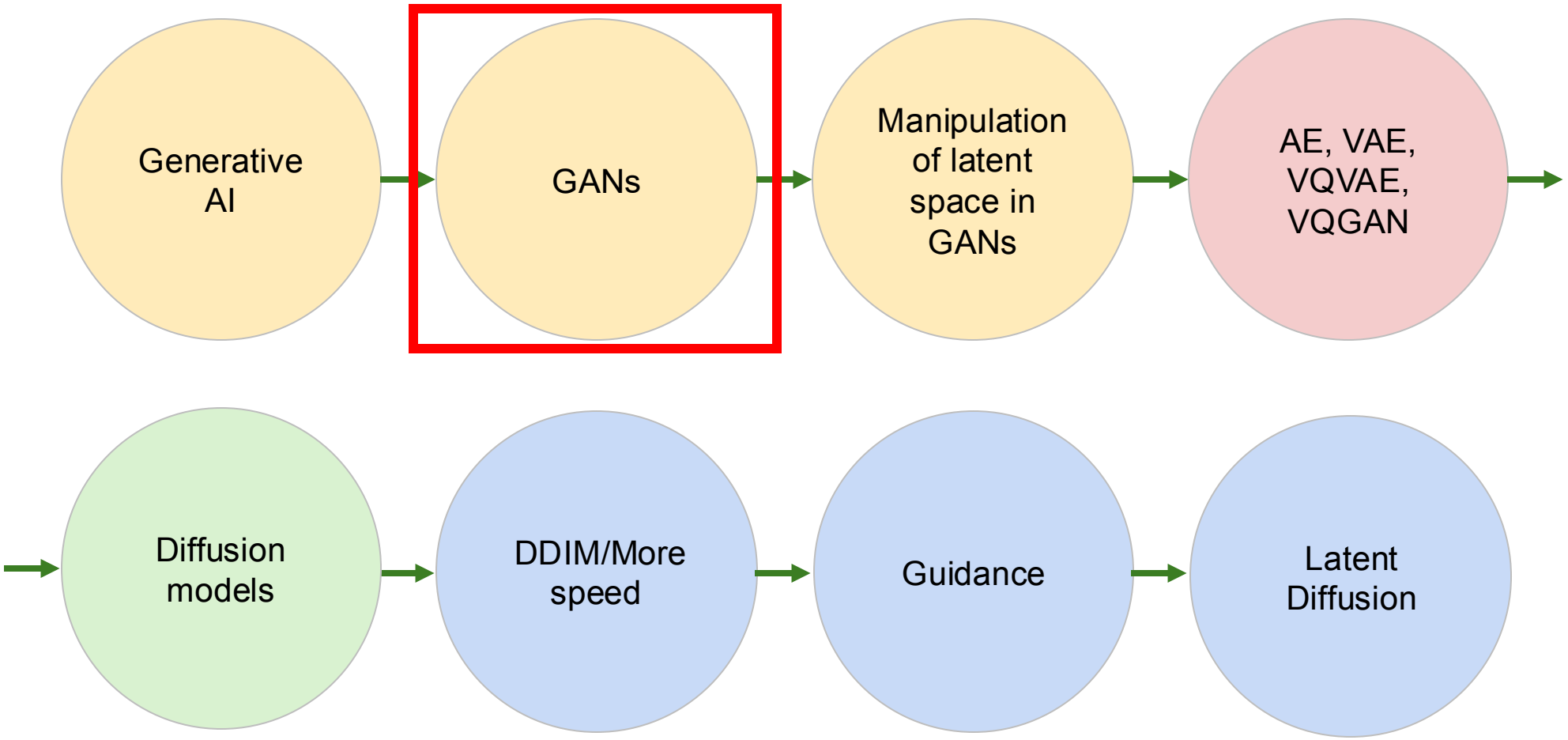
"A person punches in a manner consistent with martial arts."



"A person is skipping rope."

"a man kicks with something or someone with his left leg."

Multimodal Generative AI with Diffusion

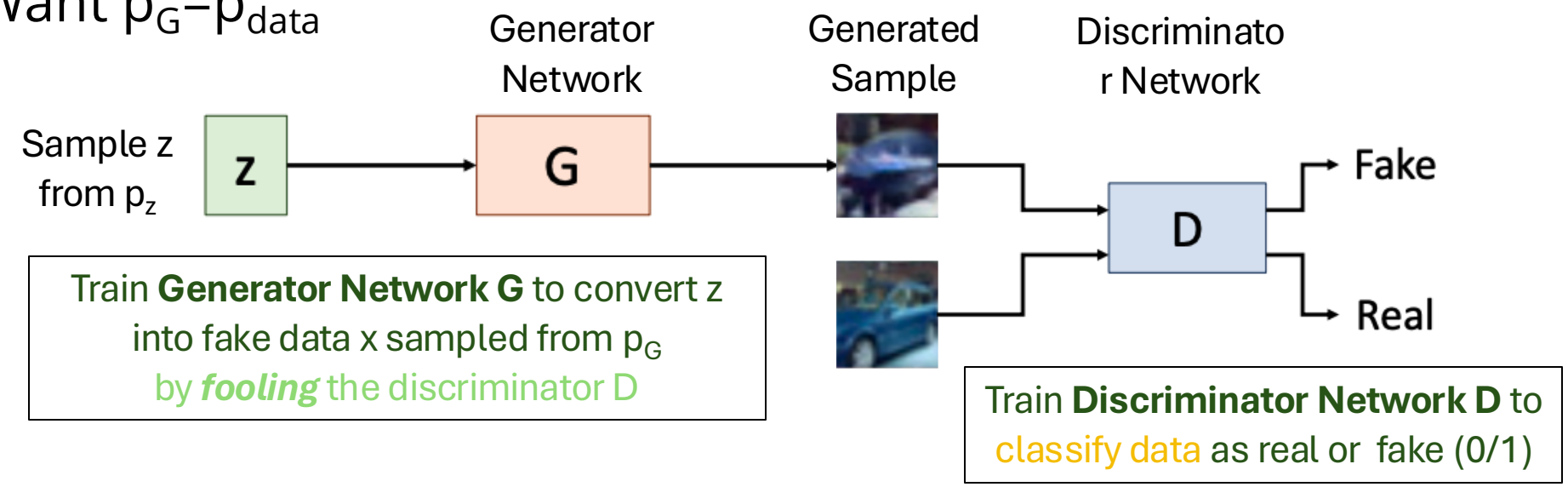


Part II: Outline

- GANs
  - Training objective
  - Training GANs
  - GAN properties
  - Examples & Metrics
  - Conditional GANs
  - Summary

Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$



GANs: Training Objective

GANs: Training Objective

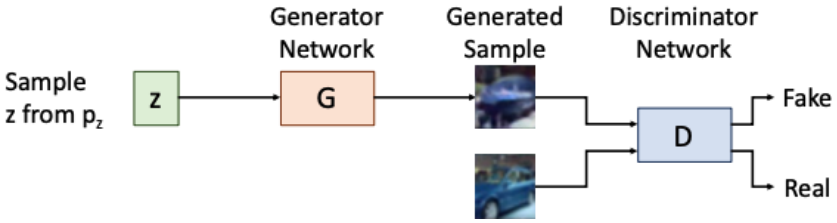
- Jointly train generator G and discriminator D with a minimax game

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

Generator wants D(x) = 1 for fake data



GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

$$= \min_{G}\max_{D} V(G, D)$$

For t in 1, … T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

$$= \min_{G}\max_{D} V(G, D)$$

We are not minimizing any overall loss! No training curves to look at!

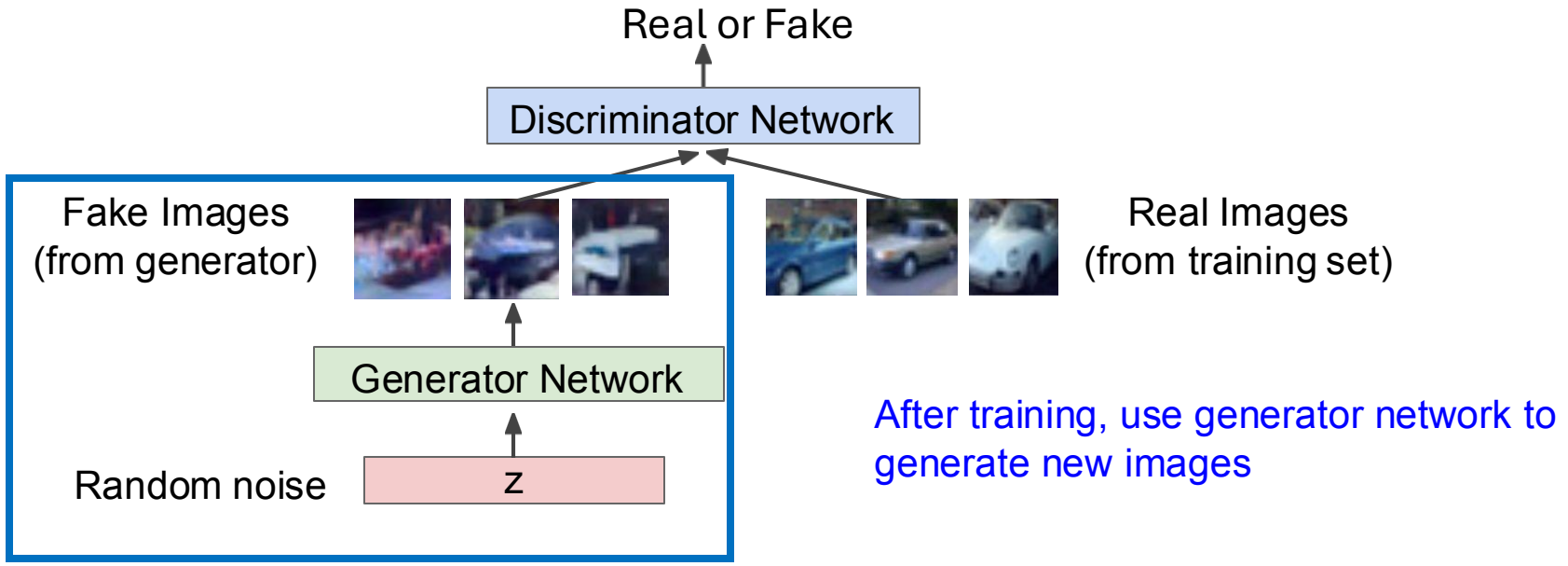
For t in 1, ... T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

Training GANs

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

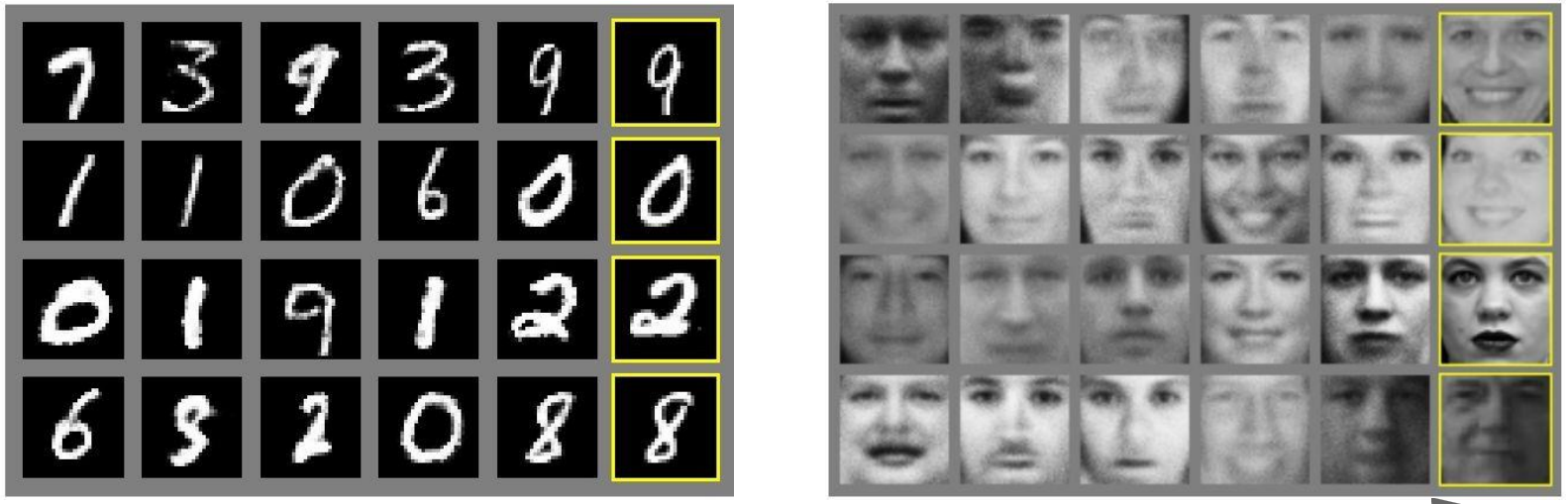


Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Examples of GANs!

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generated samples

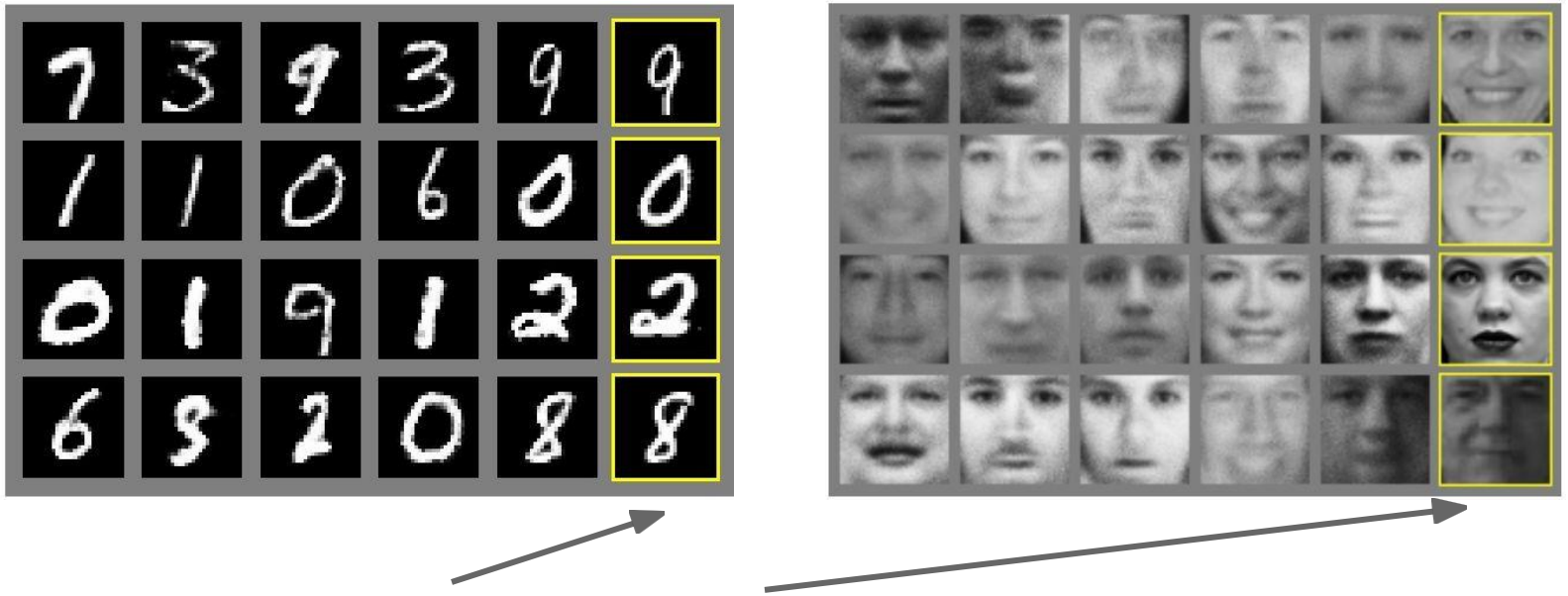


Nearest neighbor from training set

Training GANS is hard!
Mode-collapse

Generator can “memorize” real images

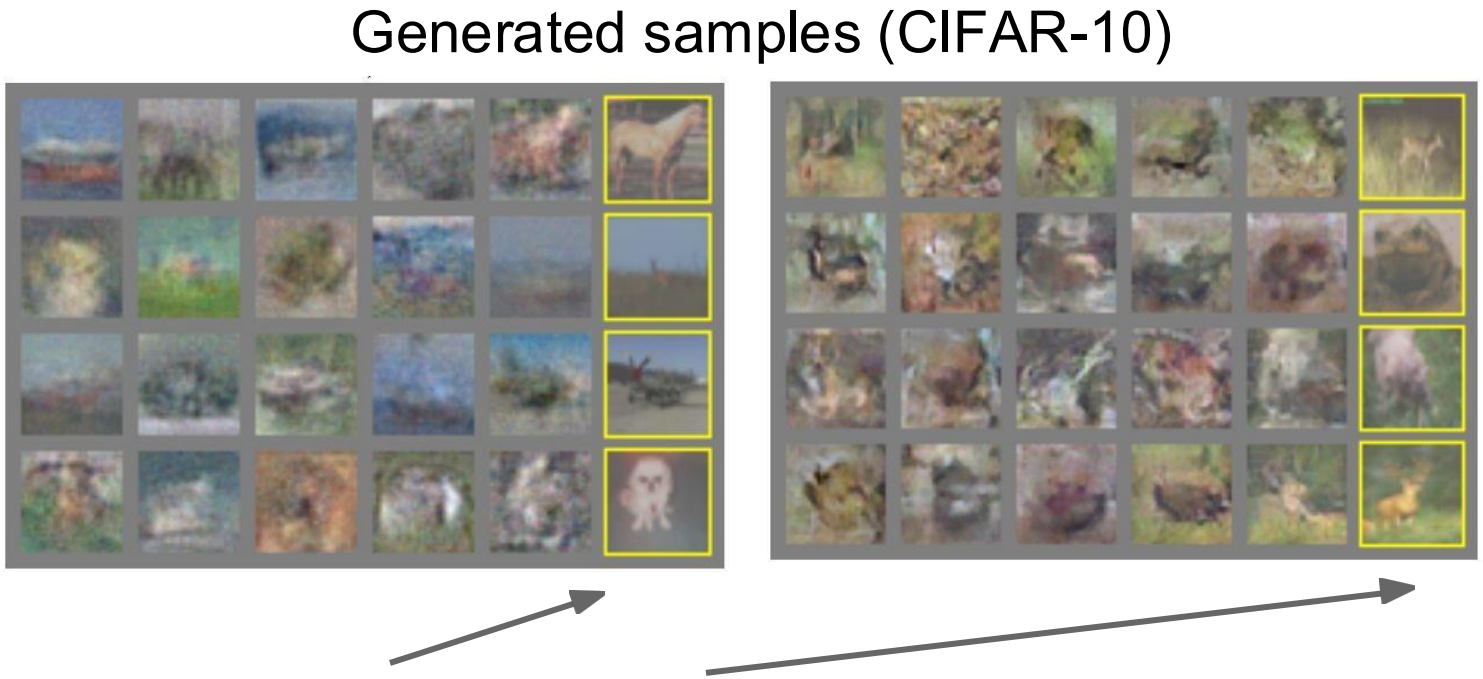
Generated samples



Nearest neighbor from training set

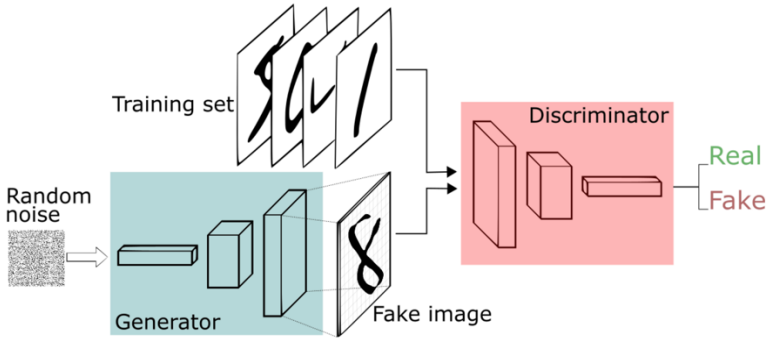
Training GANS is hard!

Generator can produce “fuzzy” images



Training GANS is hard!

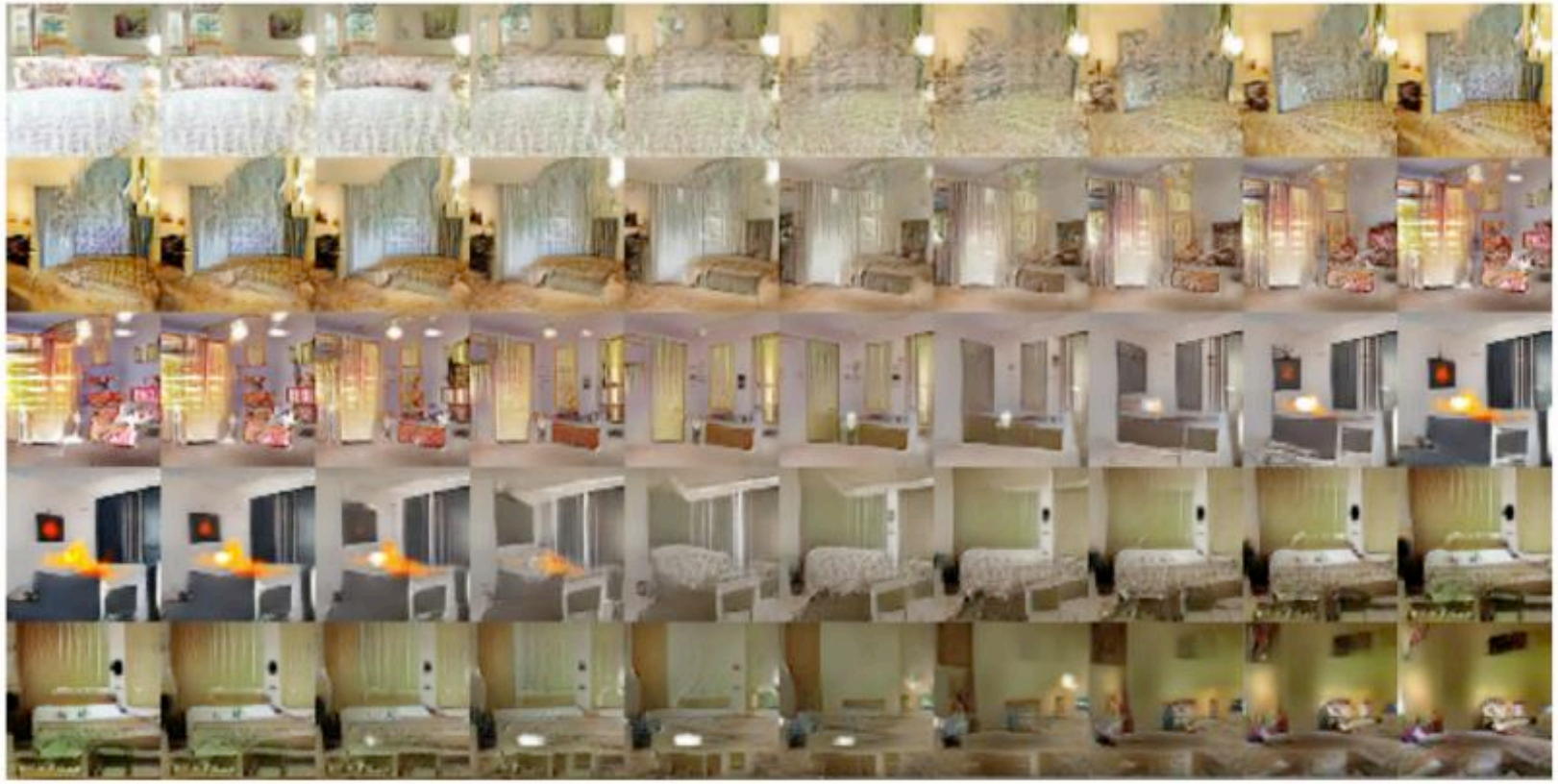
- Training GANs is still very hard
  - Many problems exist
  - Non-convergence
    - The models never converge and worse they become unstable
  - Mode collapse
    - The generator produces a single or limited modes
      - i.e. the images are not as diverse as the true data
- Many tricks exist



GAN properties

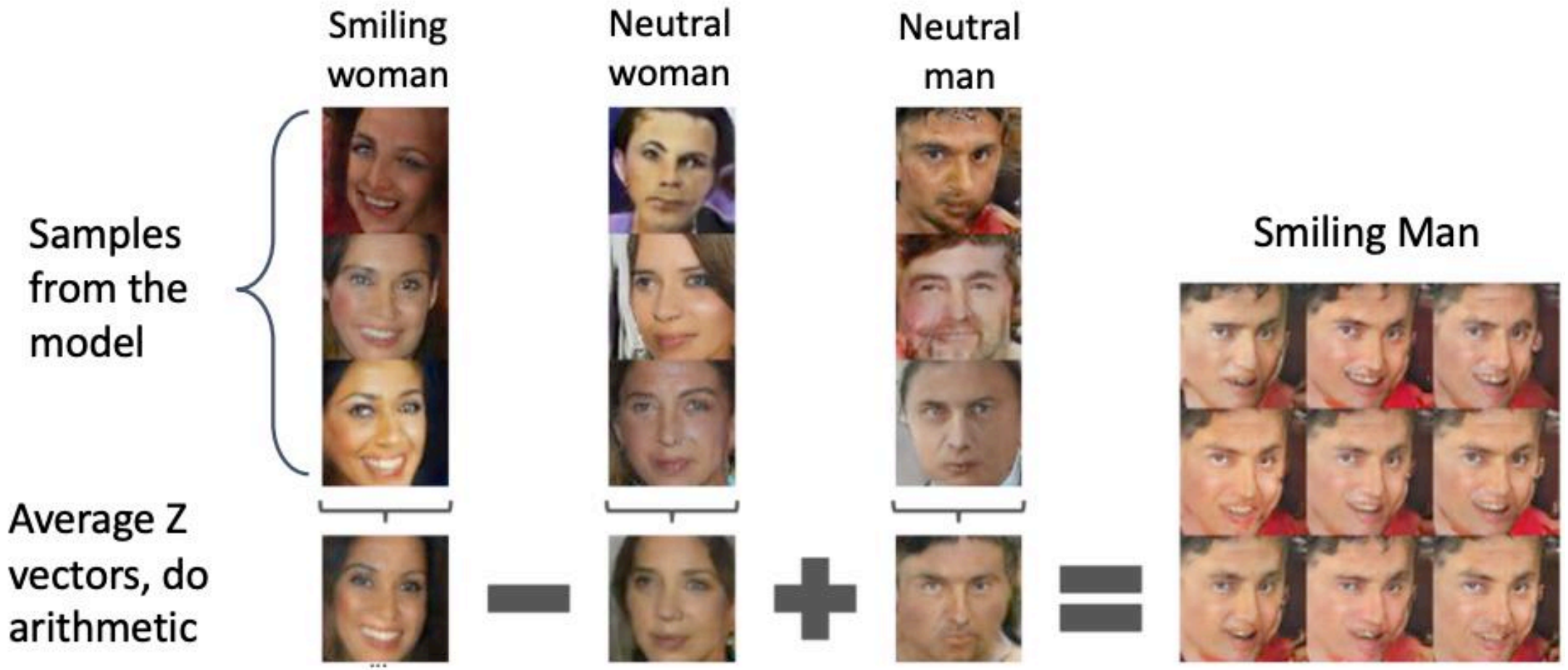
GAN Interpolation

Interpolating between points in latent z space

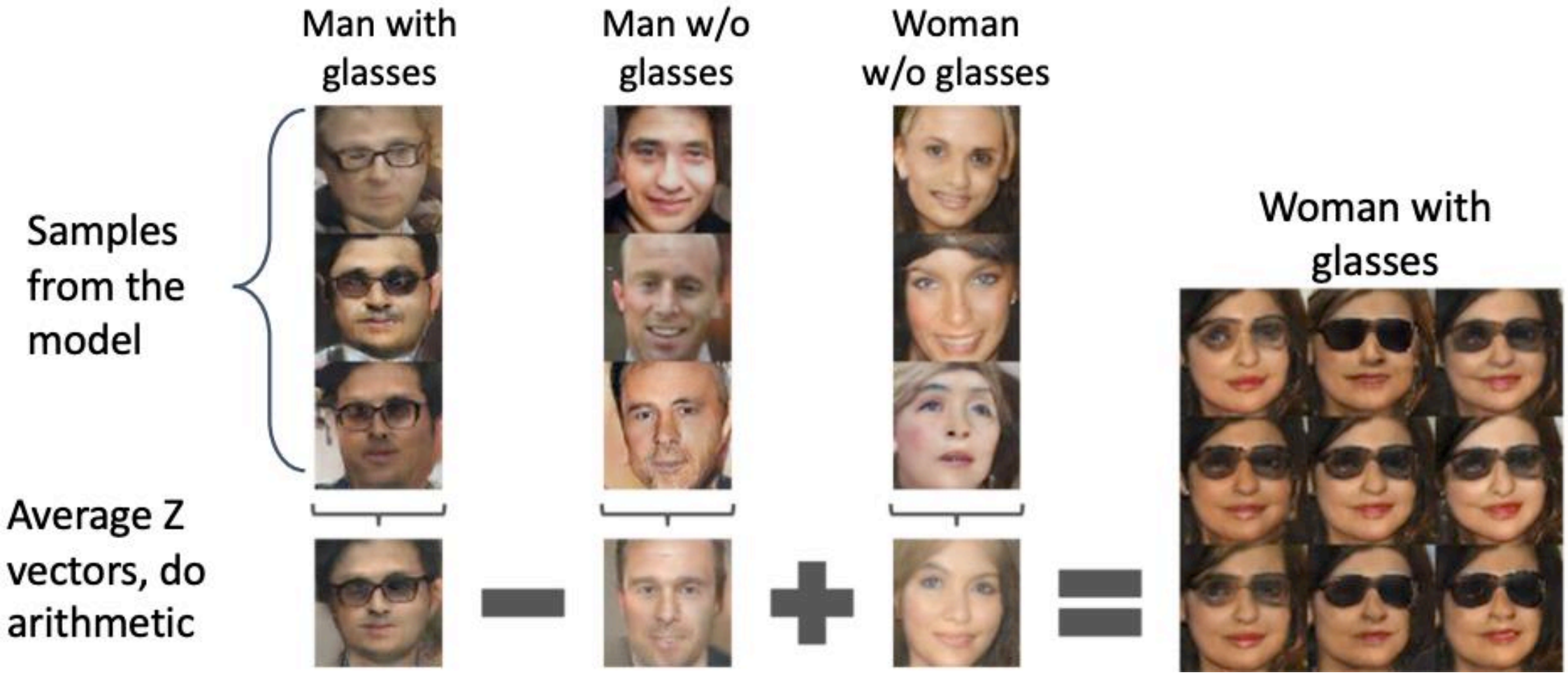


[Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." , ICLR 2016]

GAN Vector Math



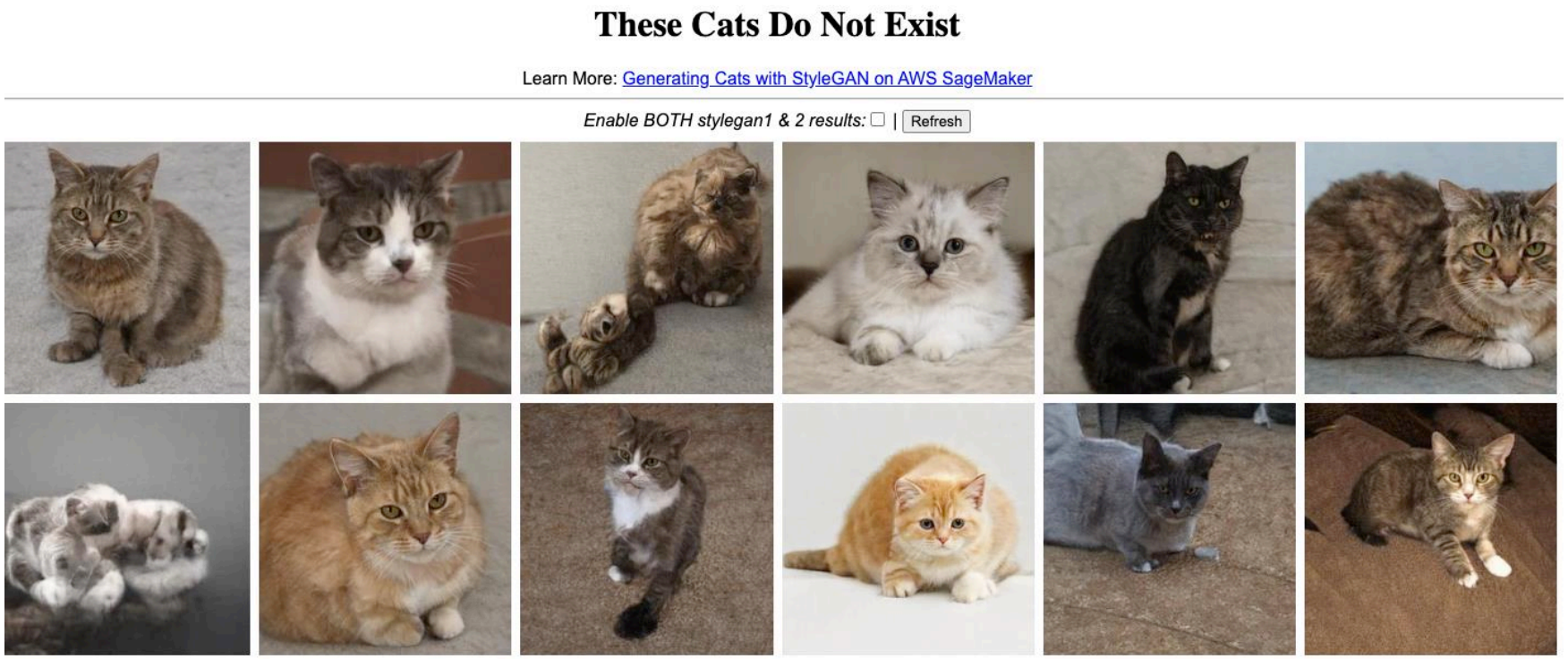
GAN Vector Math



Examples & Metrics

Examples

- https://thiscatdoesnotexist.com/



FID - Frechet Inception Distance

- A metric attempting to quantitatively evaluate the quality of generated images
- Main Idea
  - Use features learned for classification, and see how similar they are for real and generated images
- How to:
  - Take the 2048 dimensional output of the global pooling layer in Inception v3
  - Compute the mean and covariance matrix of the features for real and fake images
  - Calculate the Frechet distance between the real and generated images

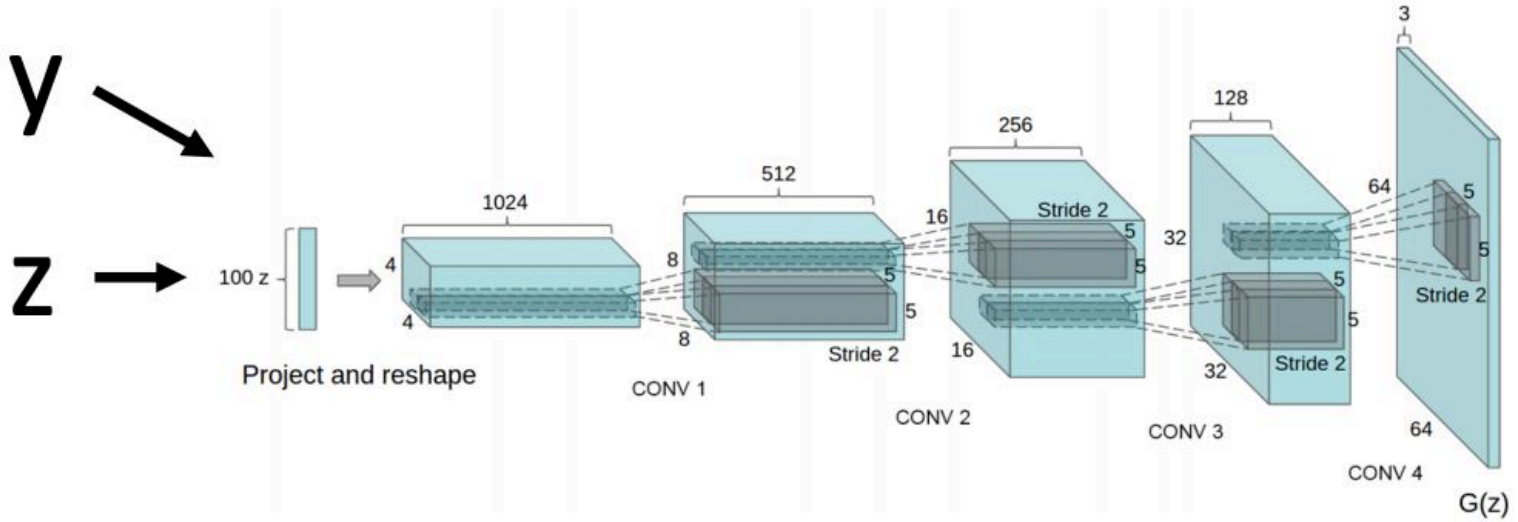
$$\|m - m_w\|_2^2 + \mathrm{Tr}\left(C + C_w - 2(CC_w)^{1/2}\right)$$

[Mercel. Martin, de al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In NeurIPS 2017]

Conditional GANs

Conditional GANs

- Conditional GAN: learn p(x|y) instead of p(x)
  - Make generator and discriminator both take label y as an additional input



Conditional GANs

Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Learn a separate scale and shift for each different label y

Conditional Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

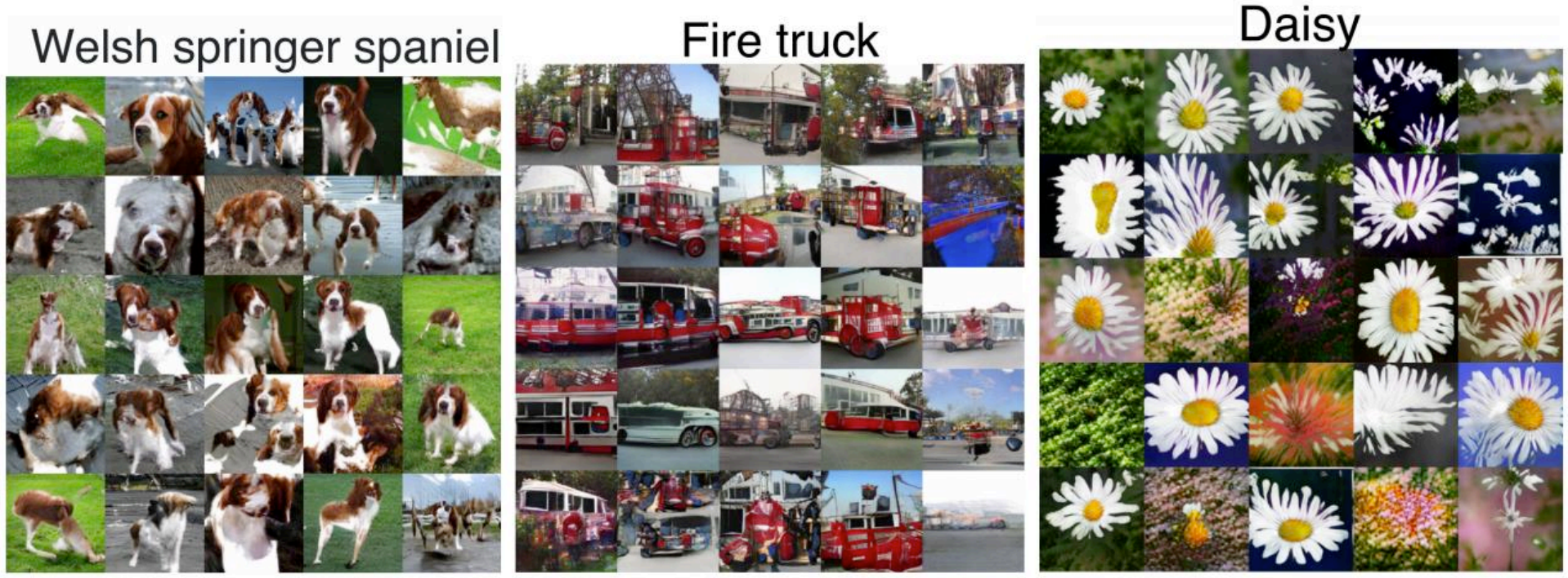
$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \boldsymbol{\gamma_j^y} \hat{x}_{i,j} + \boldsymbol{\beta_j^y}$$

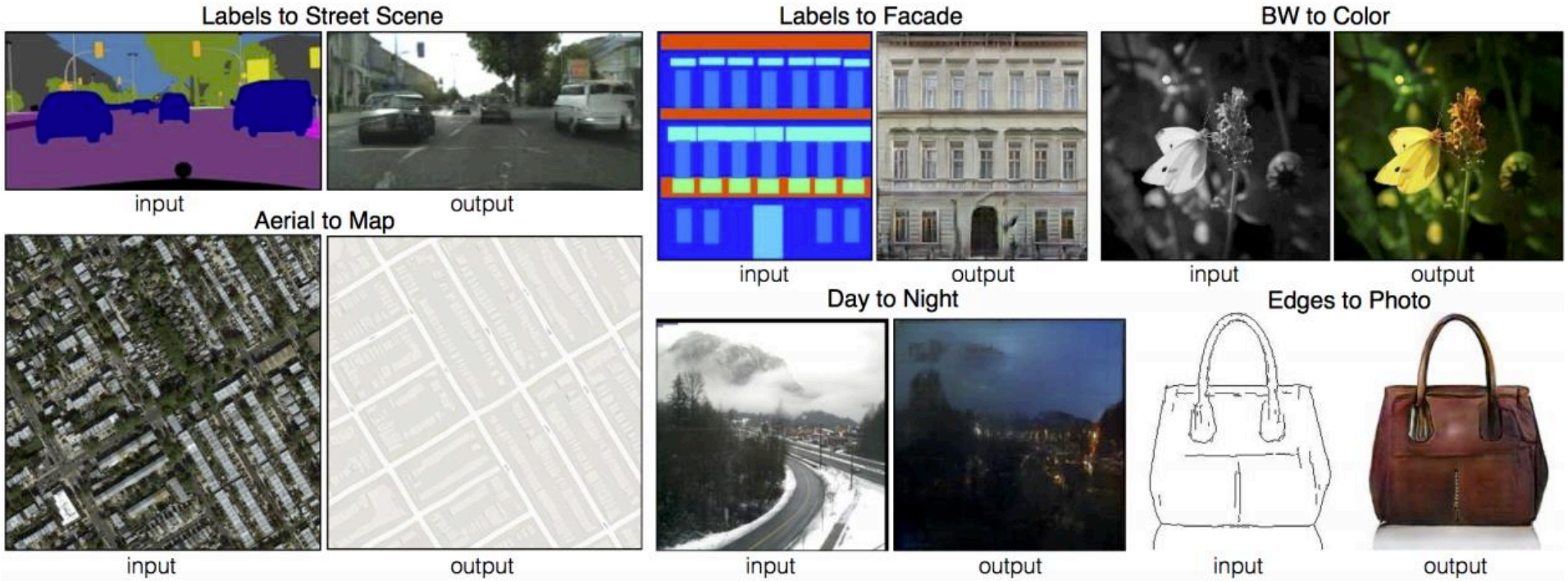
Dumoulin et al, “A learned representation for artistic style”, ICLR 2017

Conditional GANs



Miyato et al, “Spectral Normalization for Generative Adversarial Networks”, ICLR 2018

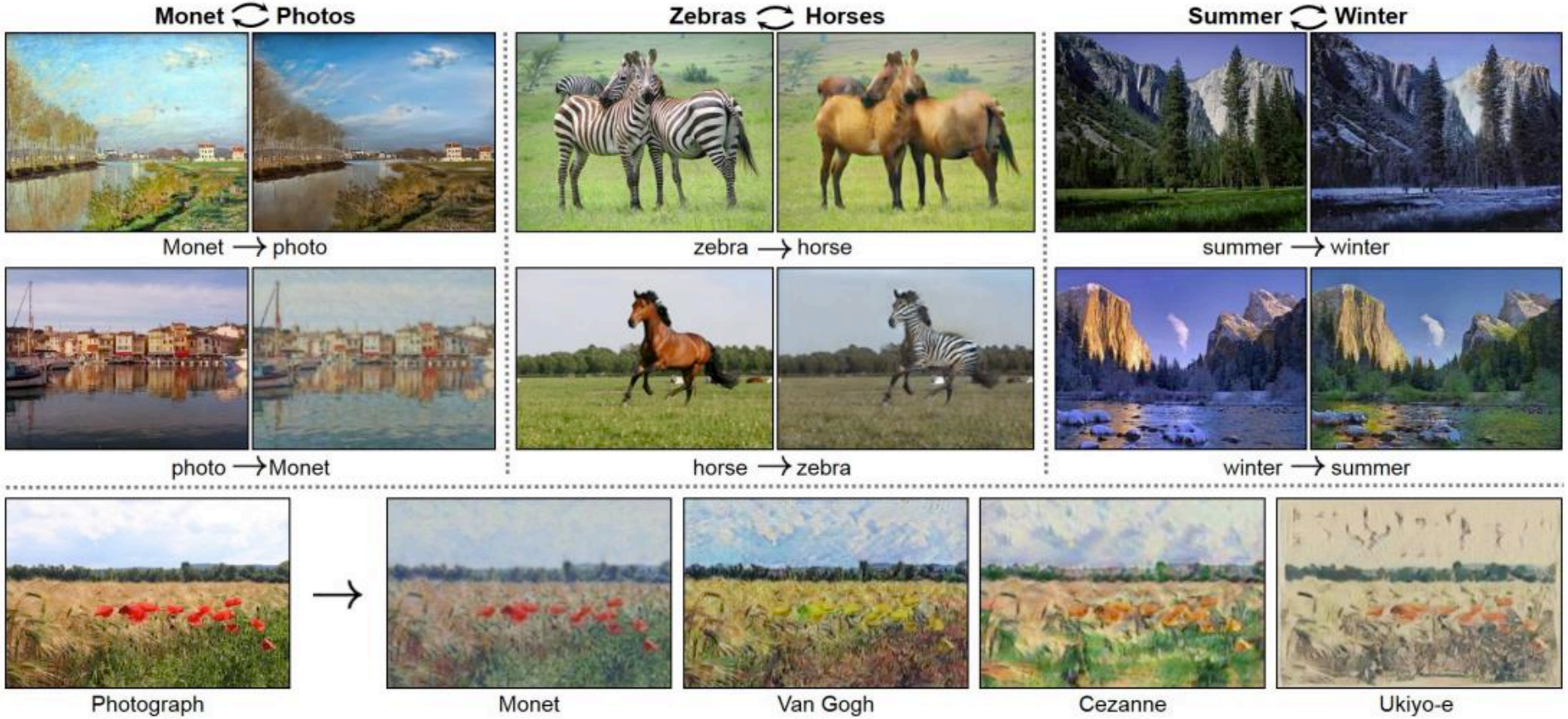
Conditional GANs
Image-to-Image Translation: Pix2Pix



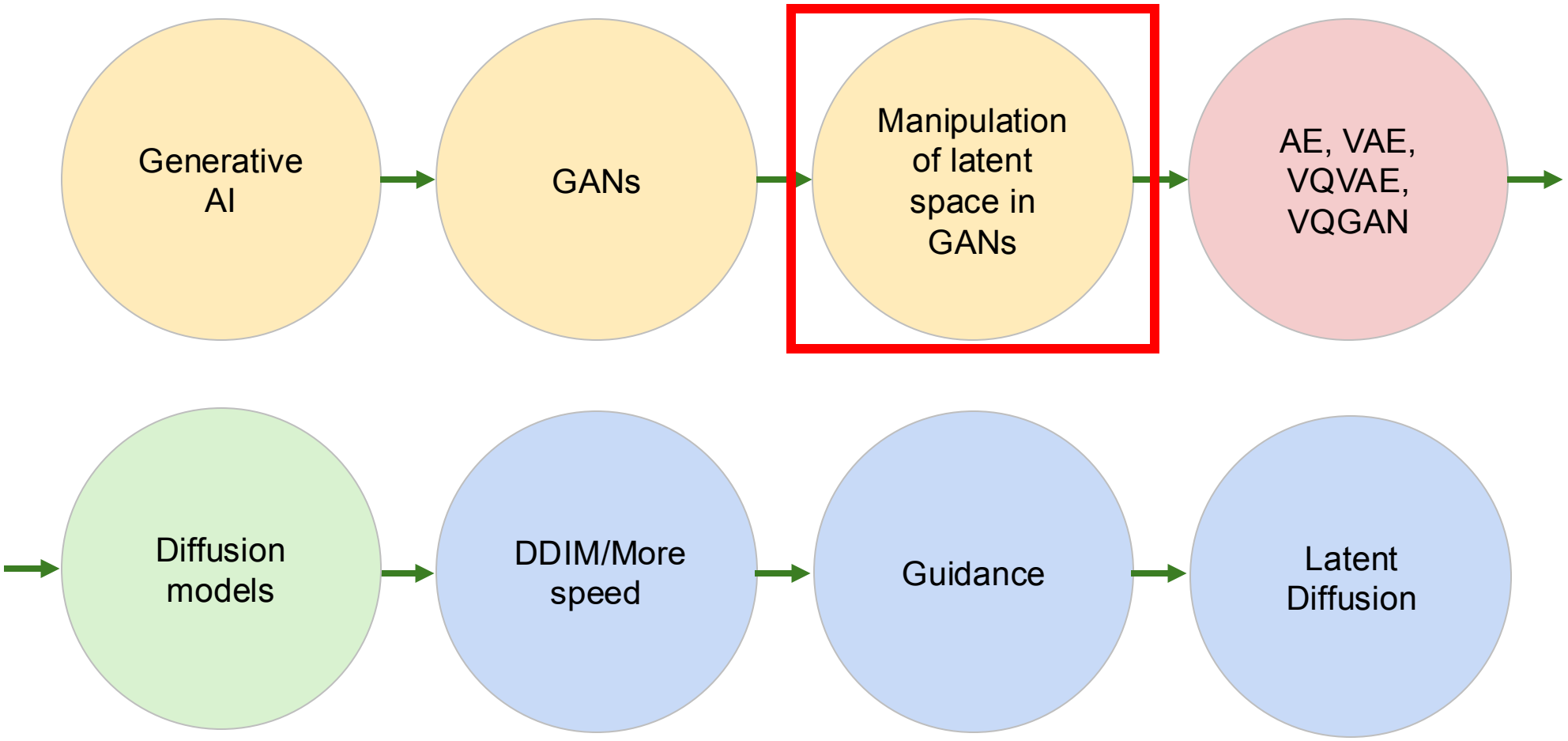
Pix2Pix

Conditional GANs
Unpaired Image-to-Image Translation: CycleGAN

CycleGAN



Multimodal Generative AI with Diffusion



Part III: Image Manipulation w GANs

- Manipulating images with GANs
  - Interpolation in latent space
  - GAN inversion
  - Learn and apply latent directions
  - CLIP + StyleGAN

Interpolation in latent space

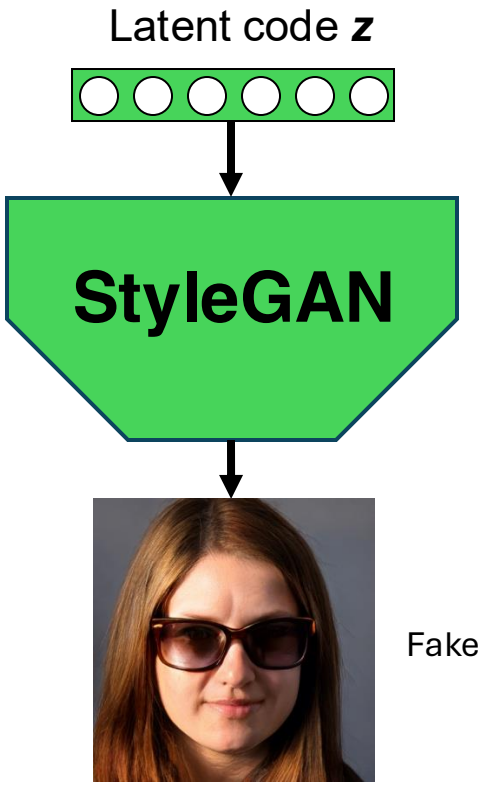
Latent space of a GAN

z_1 $a*z_1+(1-a)*z_2$ z_2

https://github.com/NVlabs/stylegan2-ada

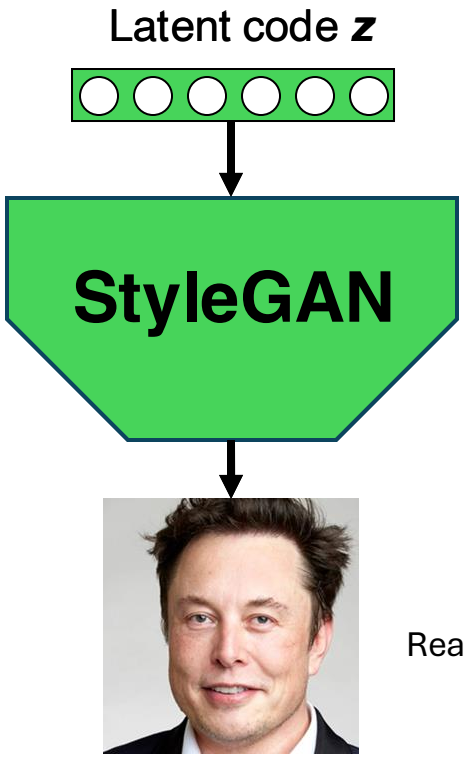
```
python generate.py --outdir=out --dlatents=out/dlatents.npz --network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl
```

Real images: GAN inversion



Fake generated image

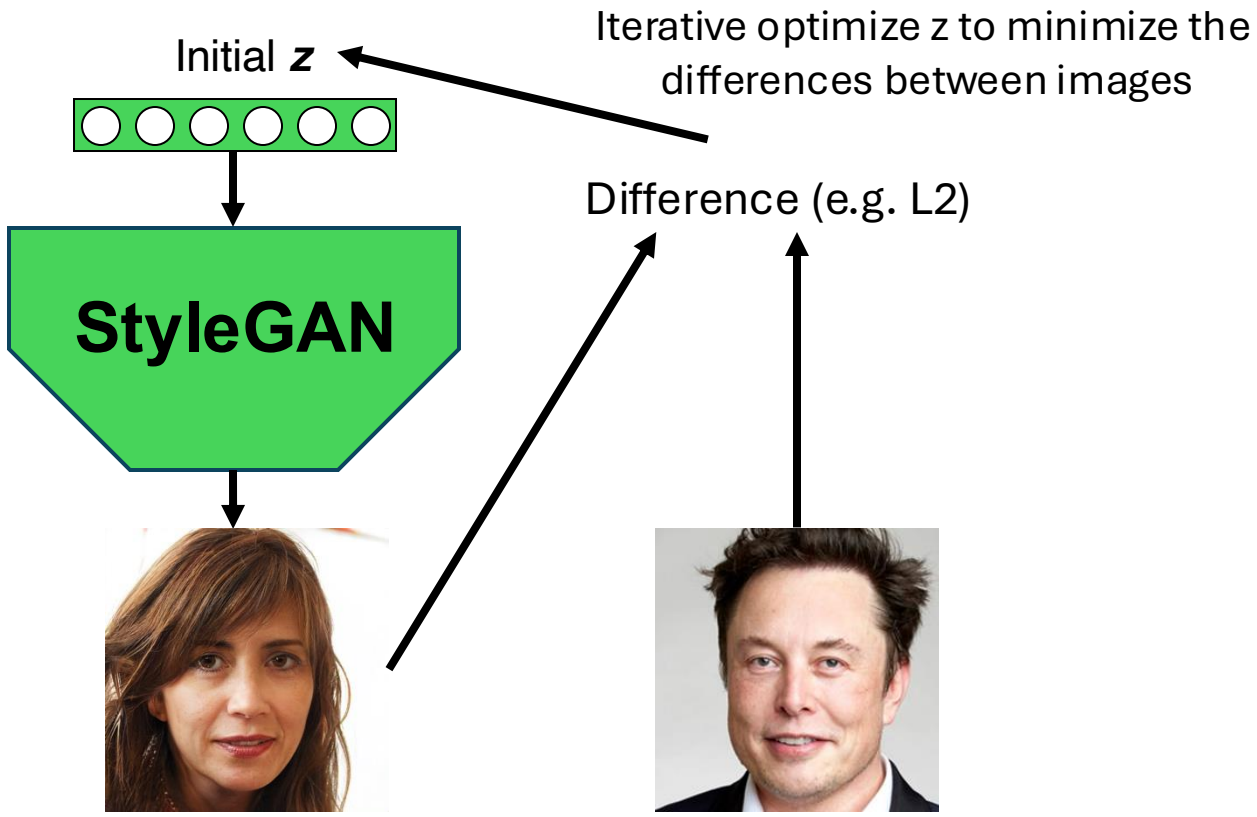
Real images: GAN inversion



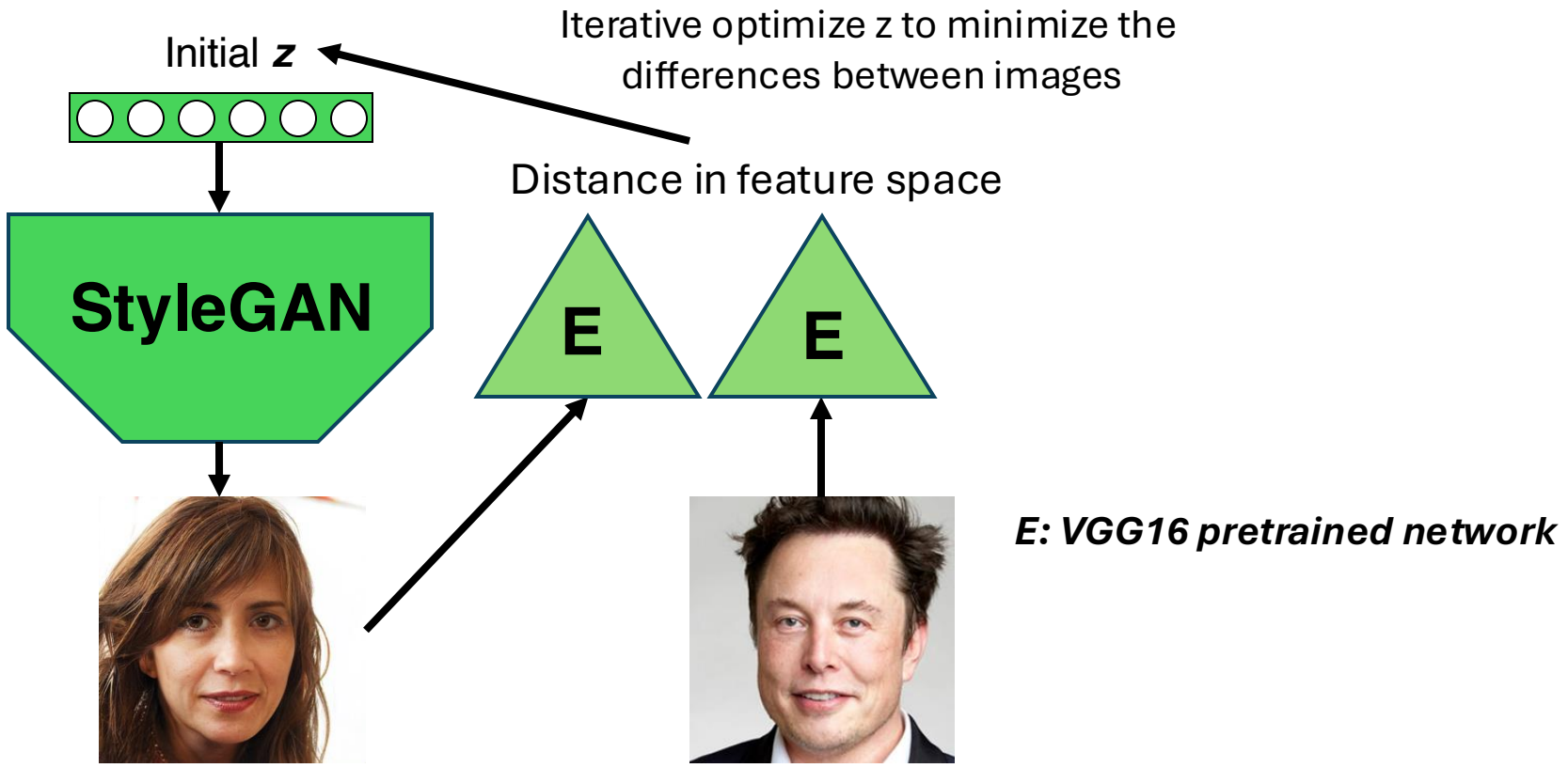
Real image

How to find z_o that generates Elon Musk?

Real images: GAN inversion



Real images: GAN inversion



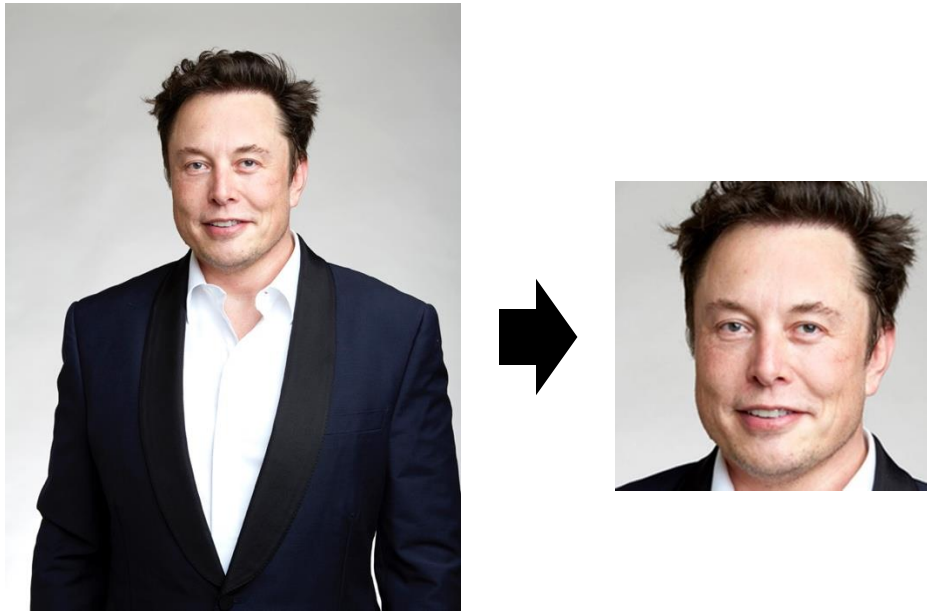
Real images: GAN inversion

```
python projector.py --outdir=out --target=targetimg.png \ --
network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl
```

Real images: GAN inversion

```
python projector.py --outdir=out --target=targetimg.png \ --
network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl
```

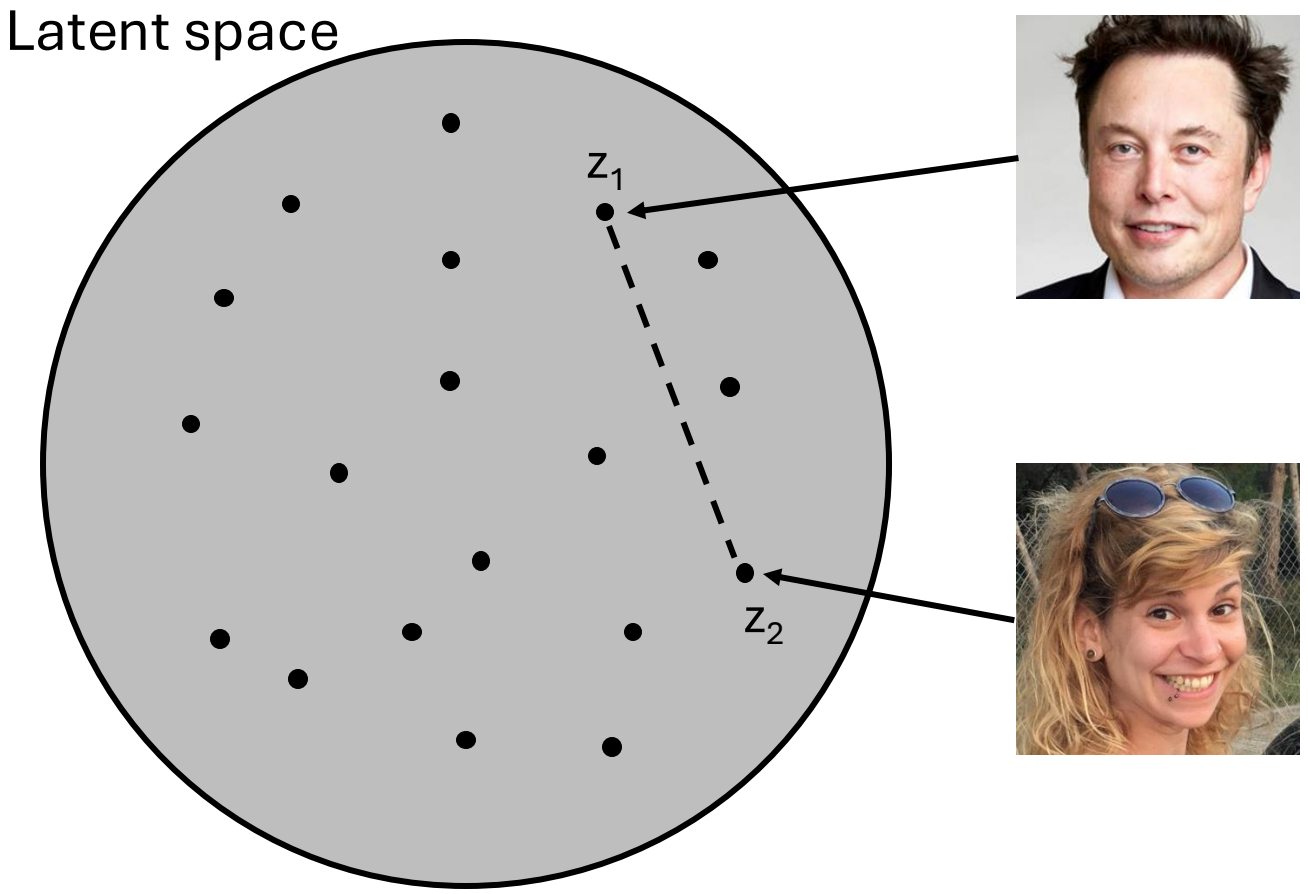
Align Images with FFHQ



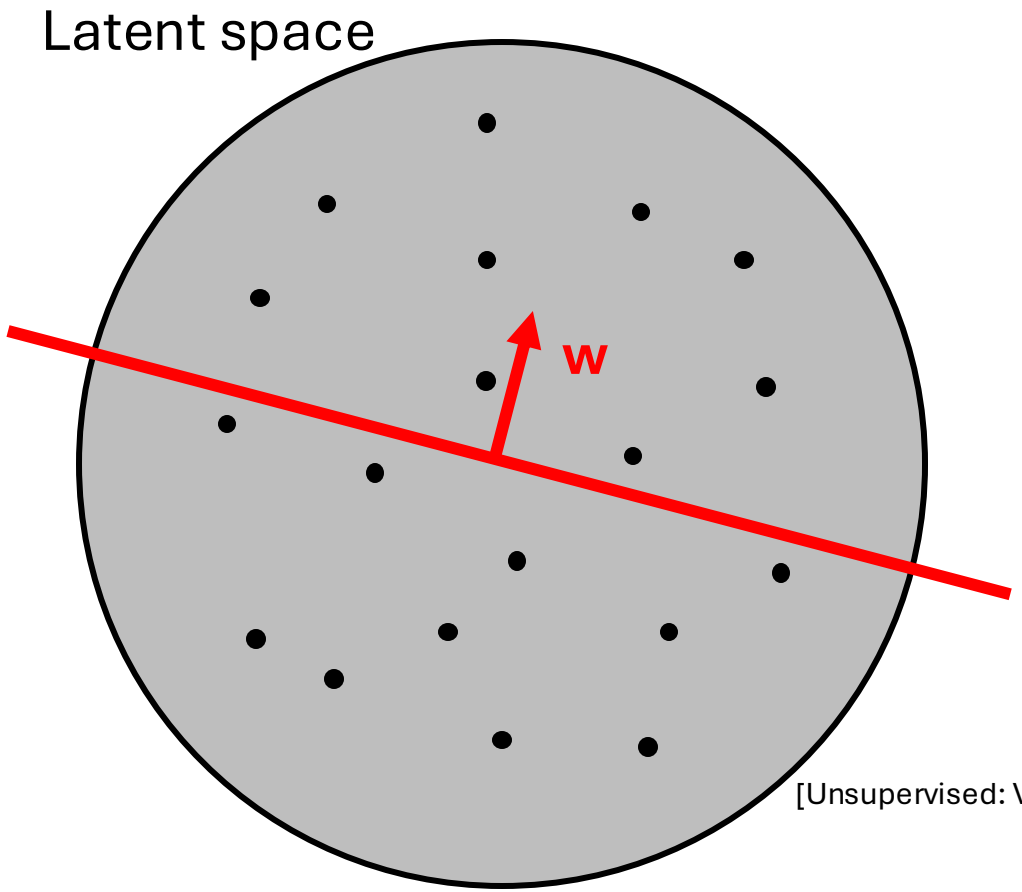
https://github.com/happy-jihye/FFHQ-Alignment

Learn and apply latent directions

Latent directions



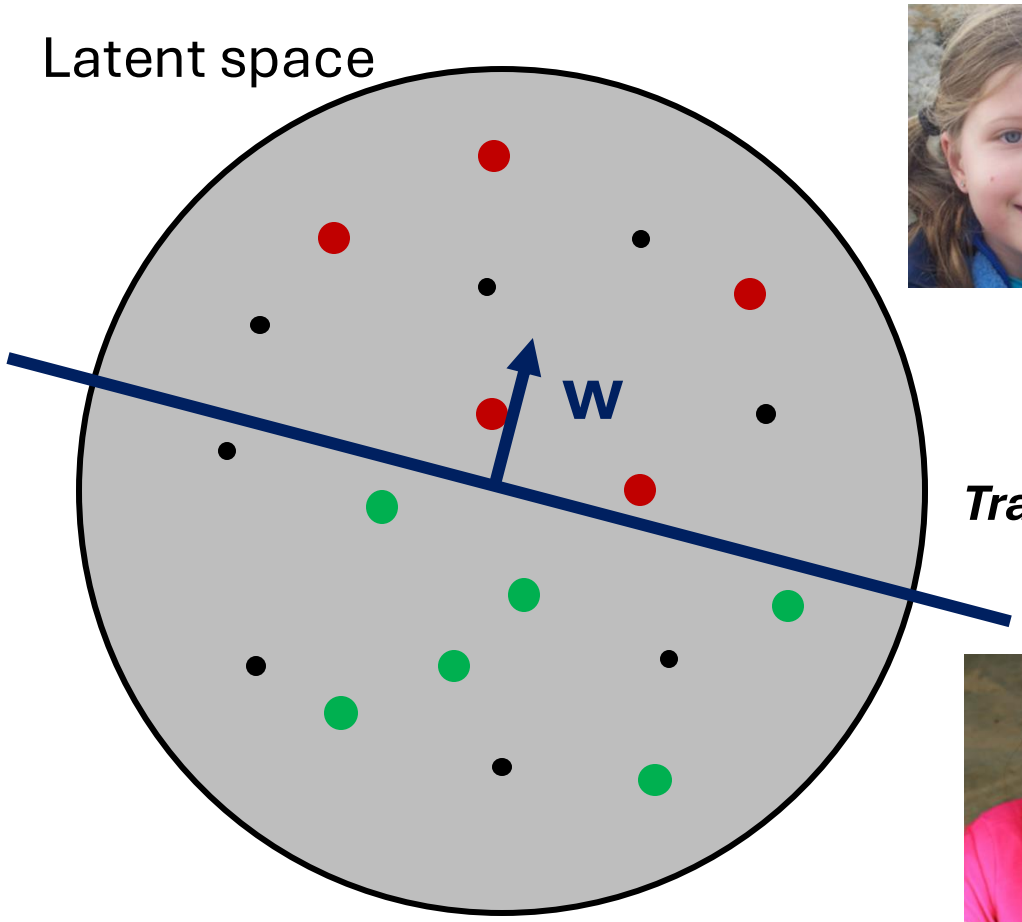
Latent directions



- Learn meaningful latent directions
- Either supervised or unsupervised
- Apply these directions in any image
- Example of directions in faces:
  - Aging
  - Smiling
  - Hair or skin color
  - Gender

[Unsupervised: Voynov and Babenko. "Unsupervised discovery of interpretable directions in the gan latent space." In ICML 2020]

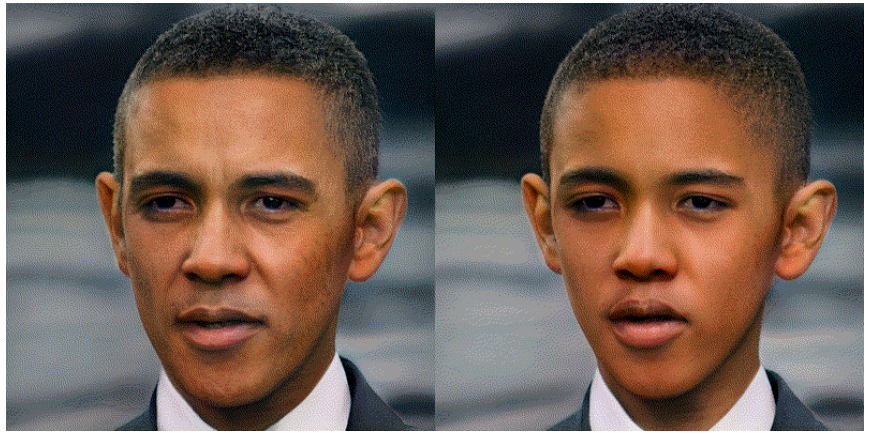
Supervised Learning of Latent directions



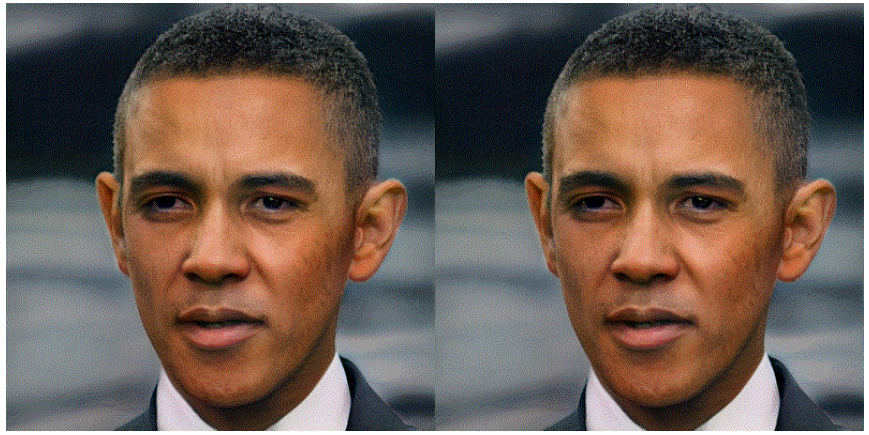
Train a linear classifier on the latent codes

Supervised Learning of Latent directions

aging



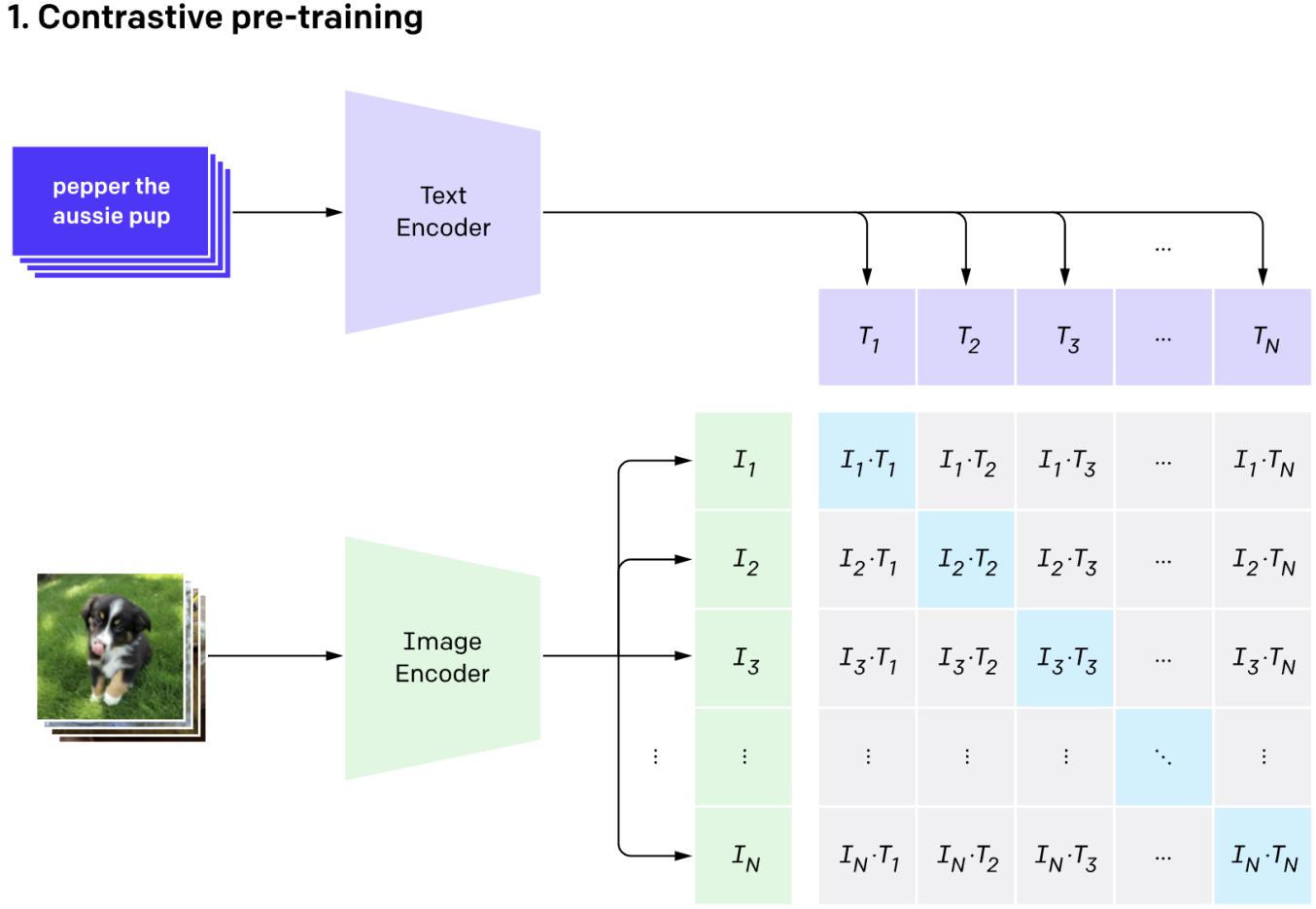
smiling



$z_{new} = z + w*m$

CLIP + StyleGAN

CLIP


1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 $I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
I_2 $I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
I_3 $I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
I_N $I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

CLIP

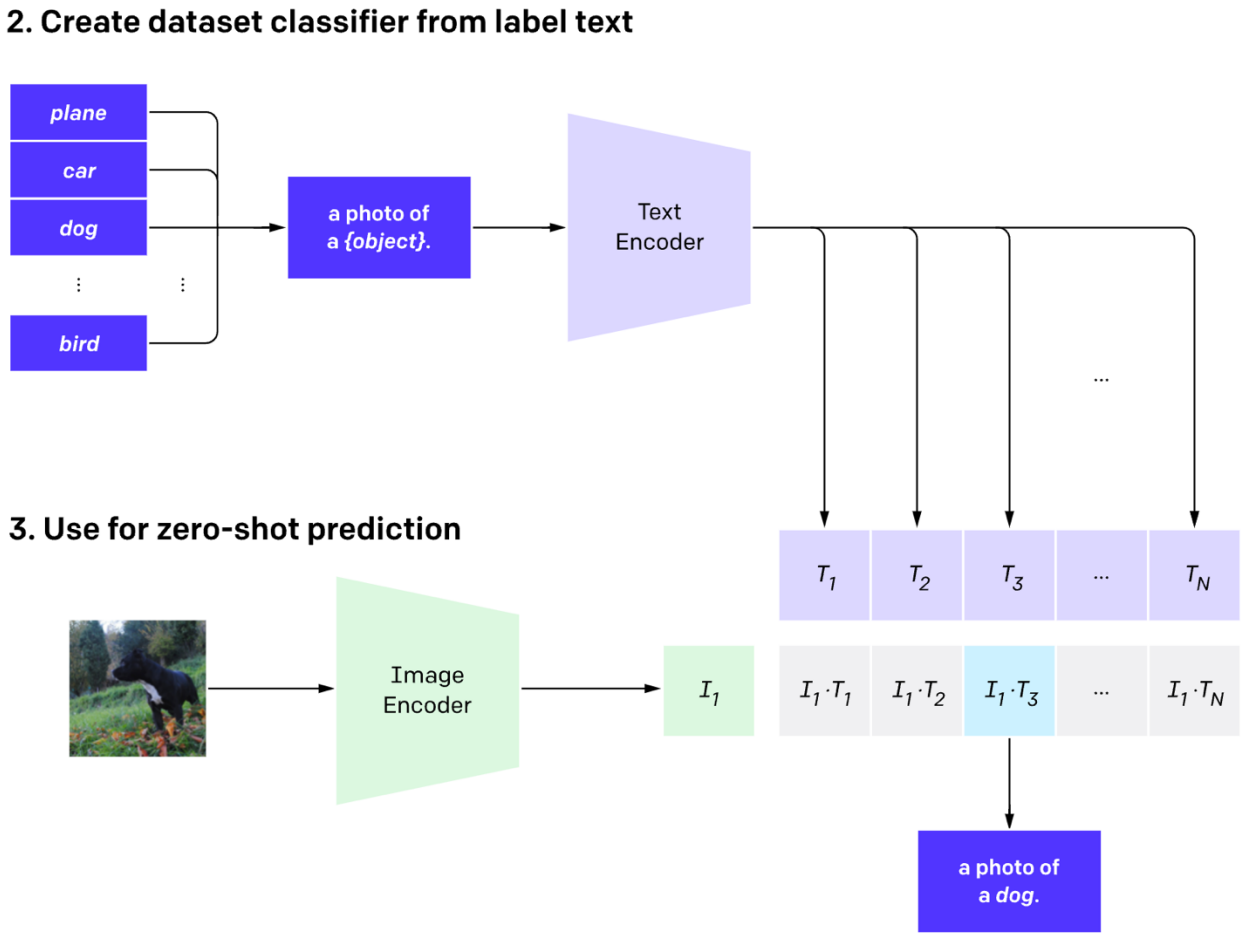

2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
...
3. Use for zero-shot prediction
T_1 T_2 T_3 ... T_N
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

Image generation from text

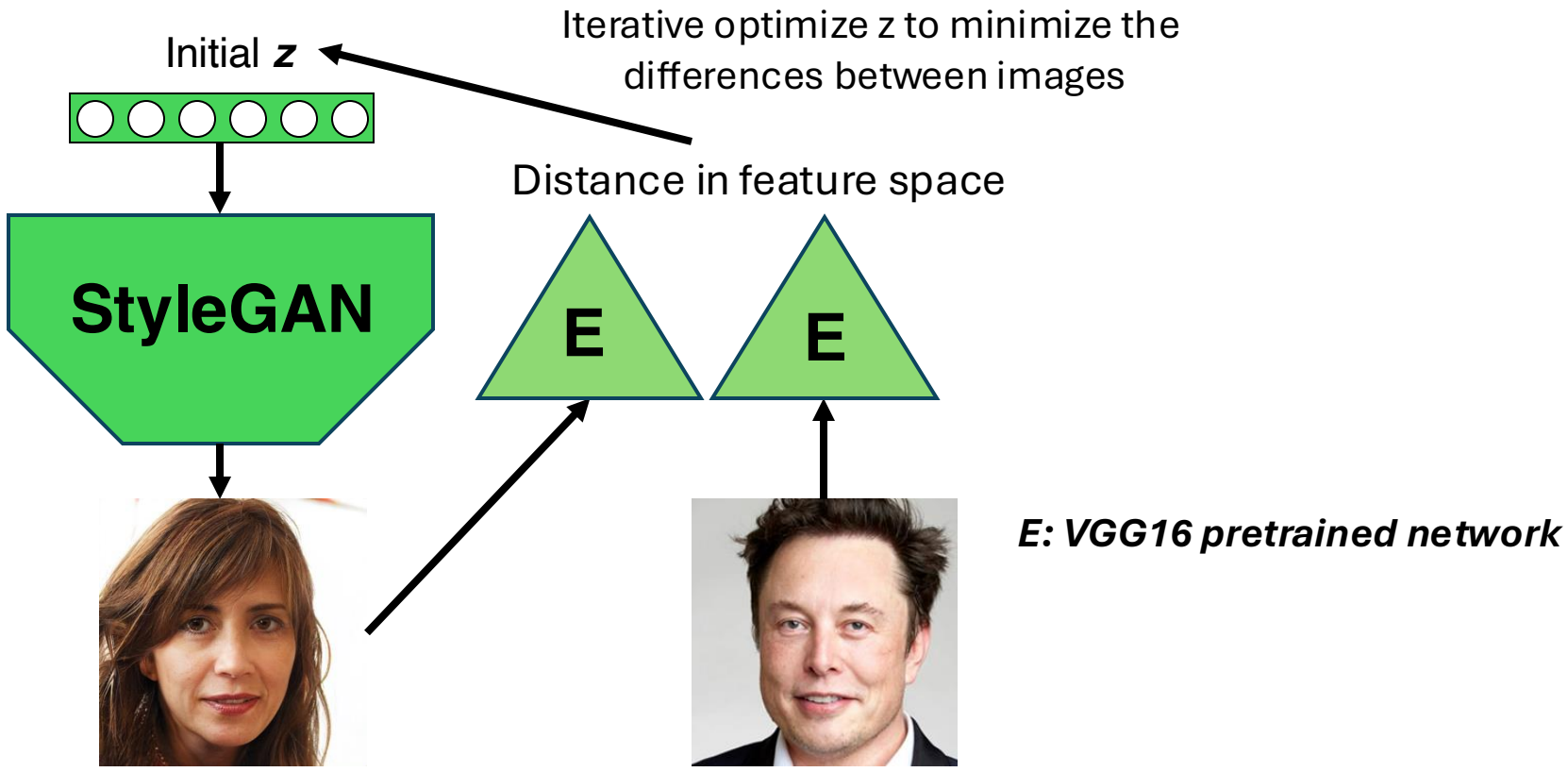
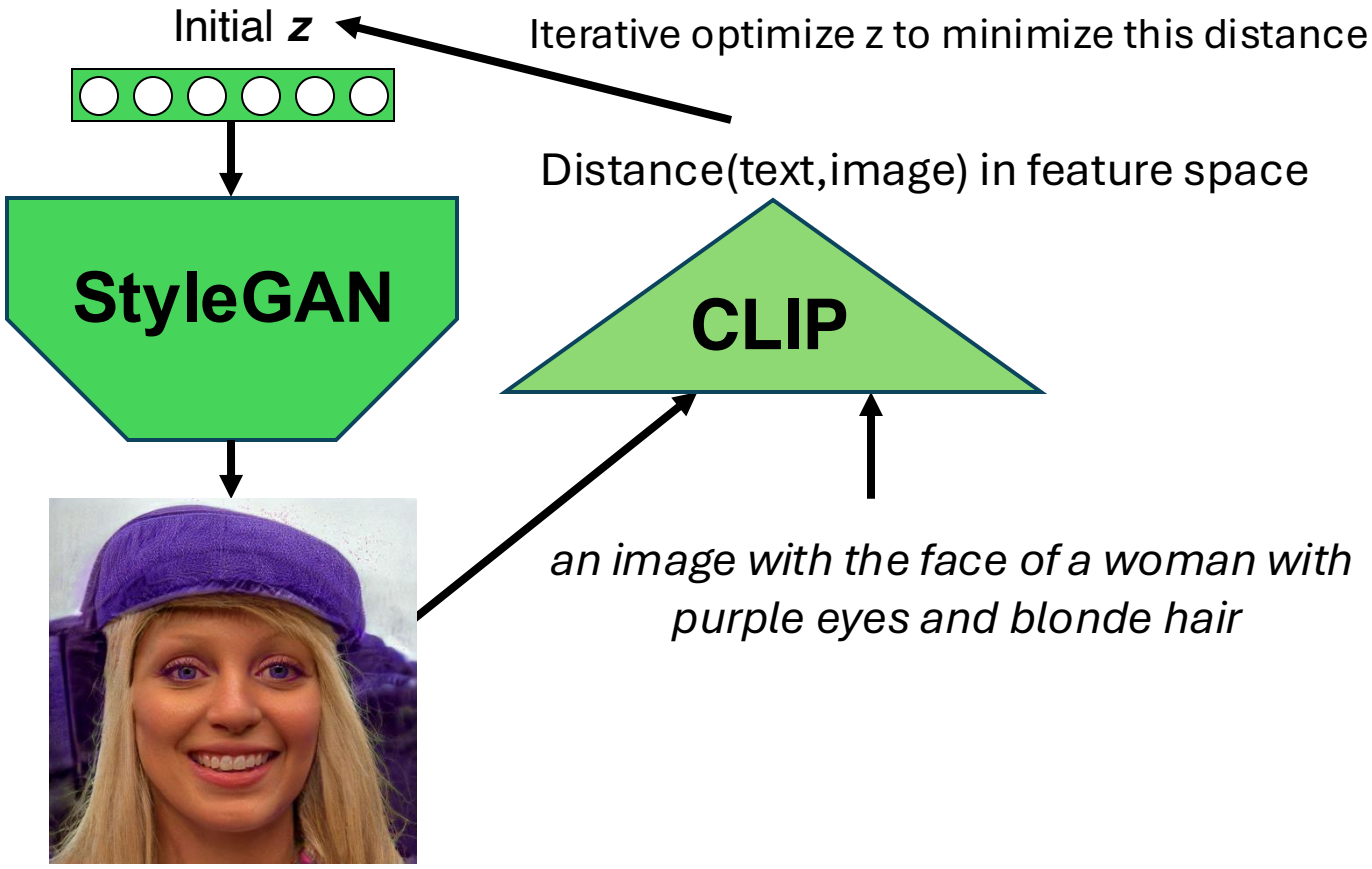
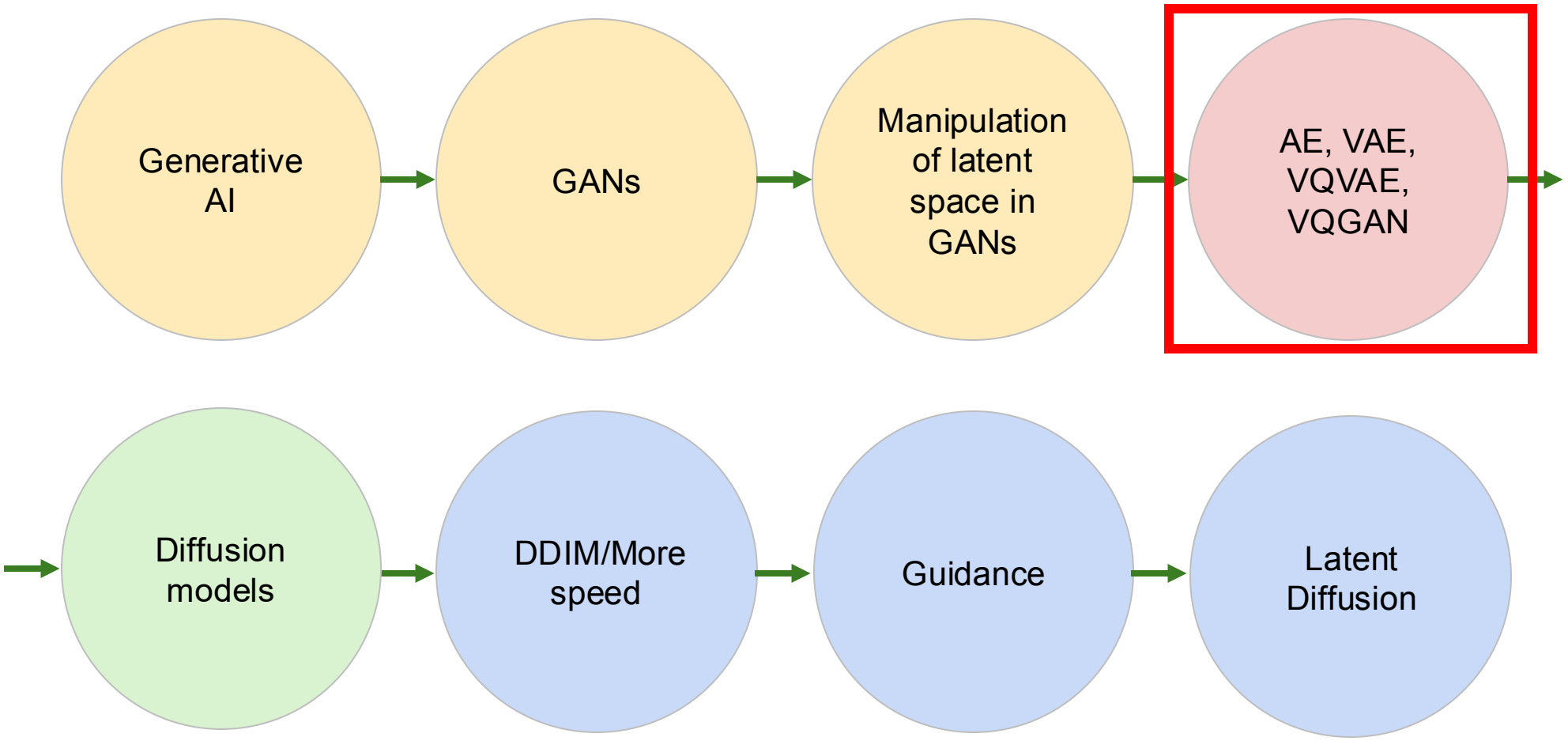


Image generation from text



Multimodal Generative AI with Diffusion



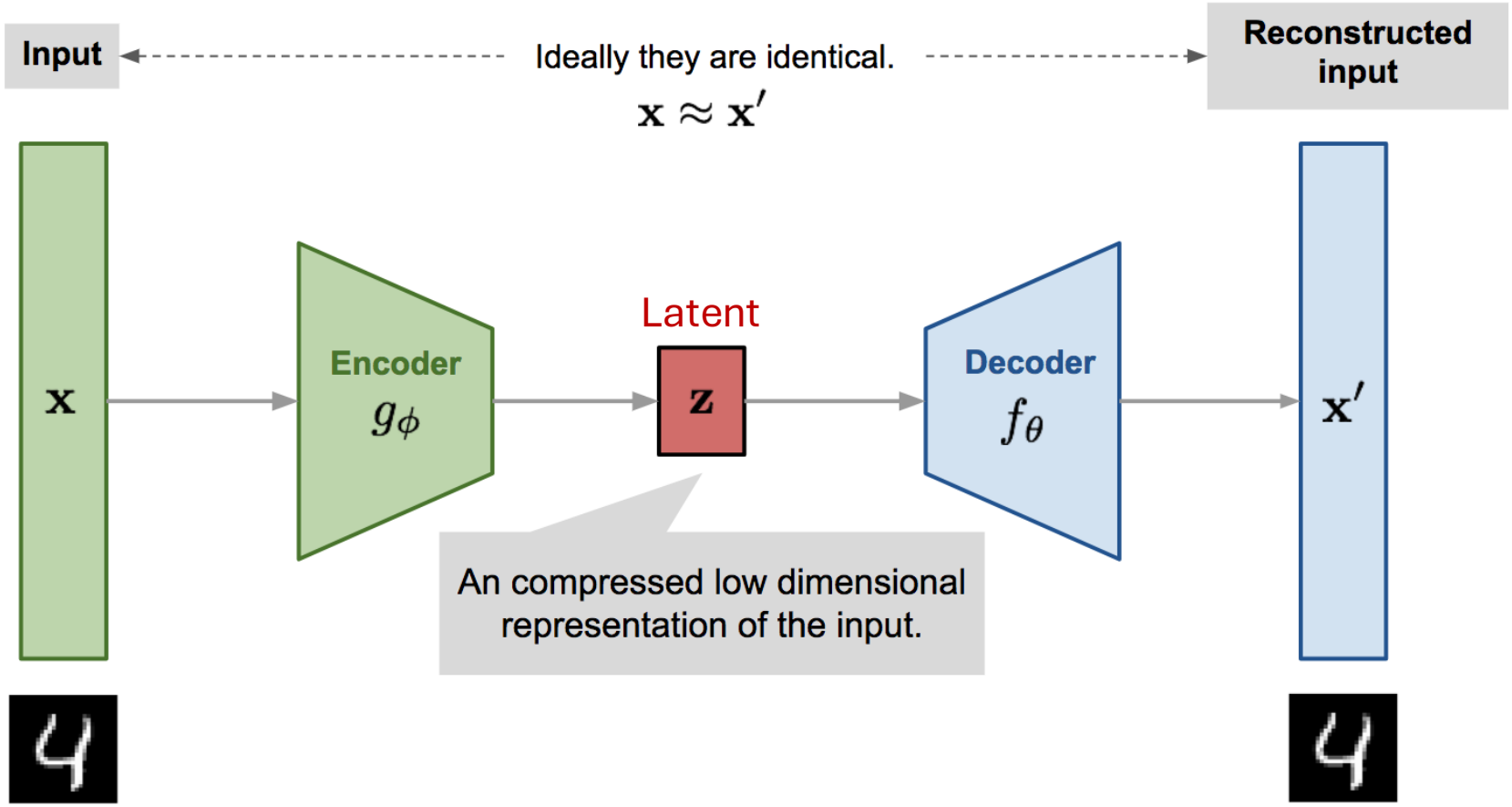
Part IV: Outline

From AutoEncoder to Vector Quantised-Variational AutoEncoder

- Introduction
- AutoEncoder (AE)
- Variational AutoEncoder (VAE)
- VAE Loss
- Parametrization Trick
- Problem with VAE
- Vector Quantised-Variational AutoEncoder (VQVAE)
- Training VQVAE
- VQGAN

AutoEncoder

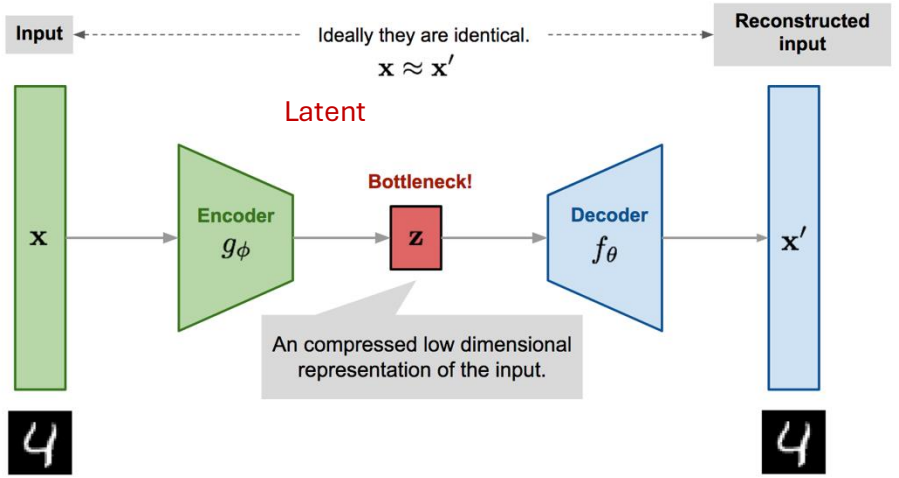
AutoEncoder (AE)



AutoEncoder (AE)

$$L_{\text{AE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)} - f_\theta \left(g_\phi \left(\mathbf{x}^{(i)} \right) \right) \right)^2$$

Reconstruction Loss



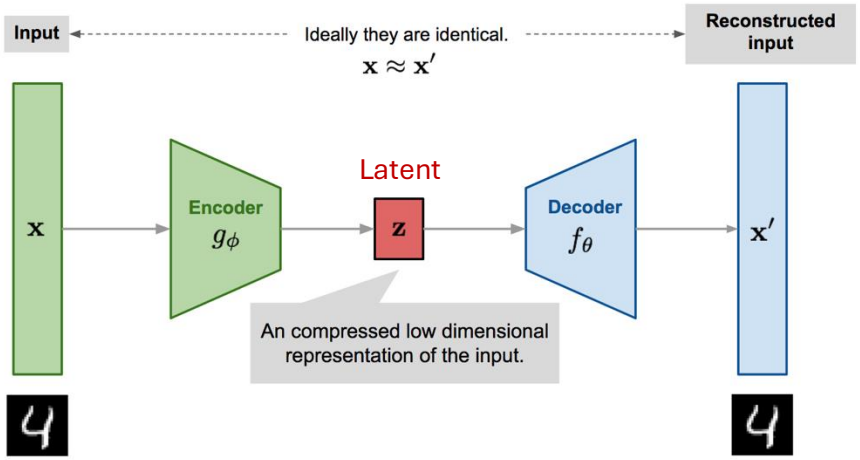
$g_\phi(.)$ The **encoding** function parameterized by ϕ.

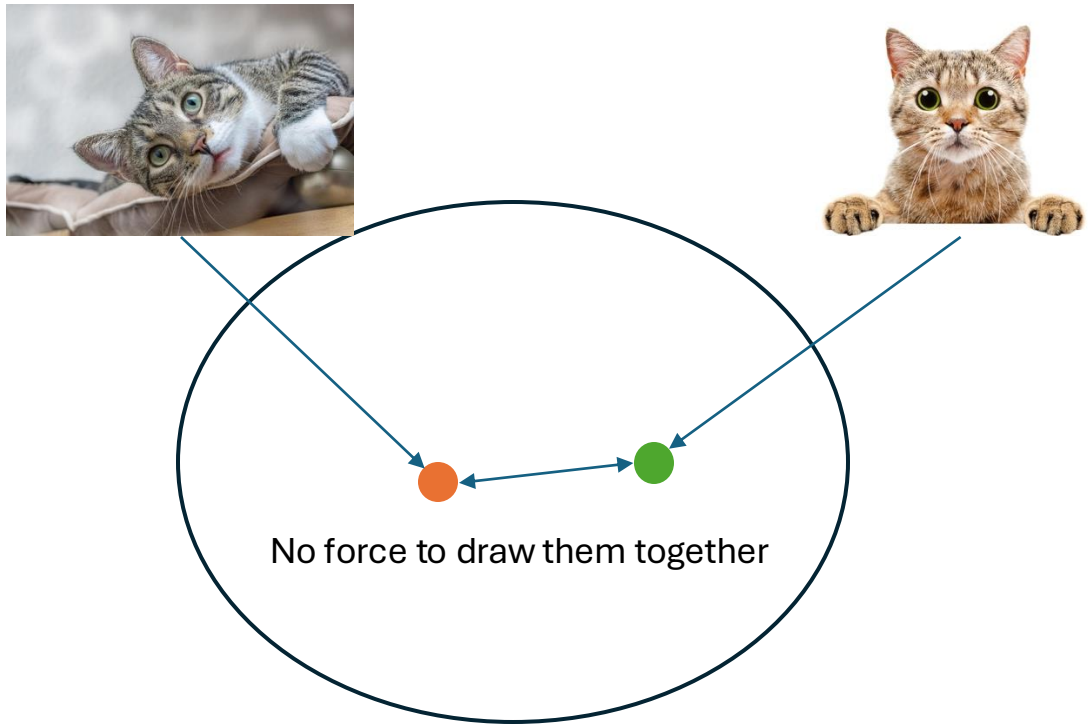
$f_\theta(.)$ The **decoding** function parameterized by θ.

$\mathbf{x}^{(i)}$ Each data point is a vector of $\boldsymbol{d}$ dimensions, $\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)} \right]$.

Hinton & Salakhutdinov, 2006

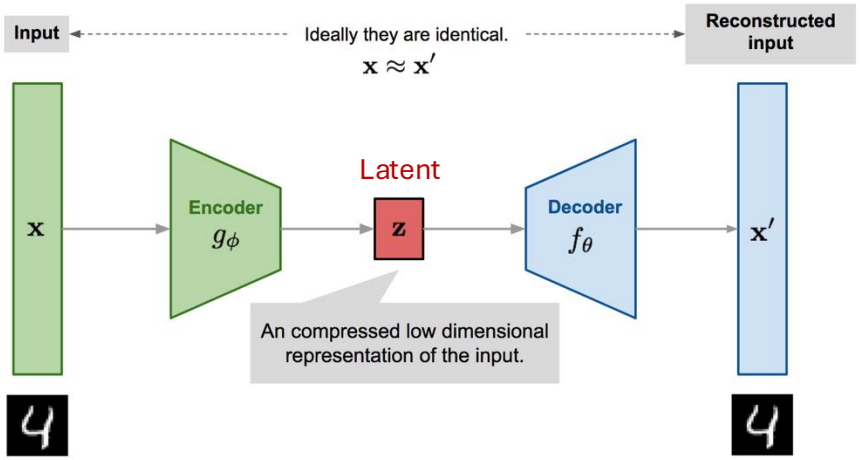
AutoEncoder (AE)

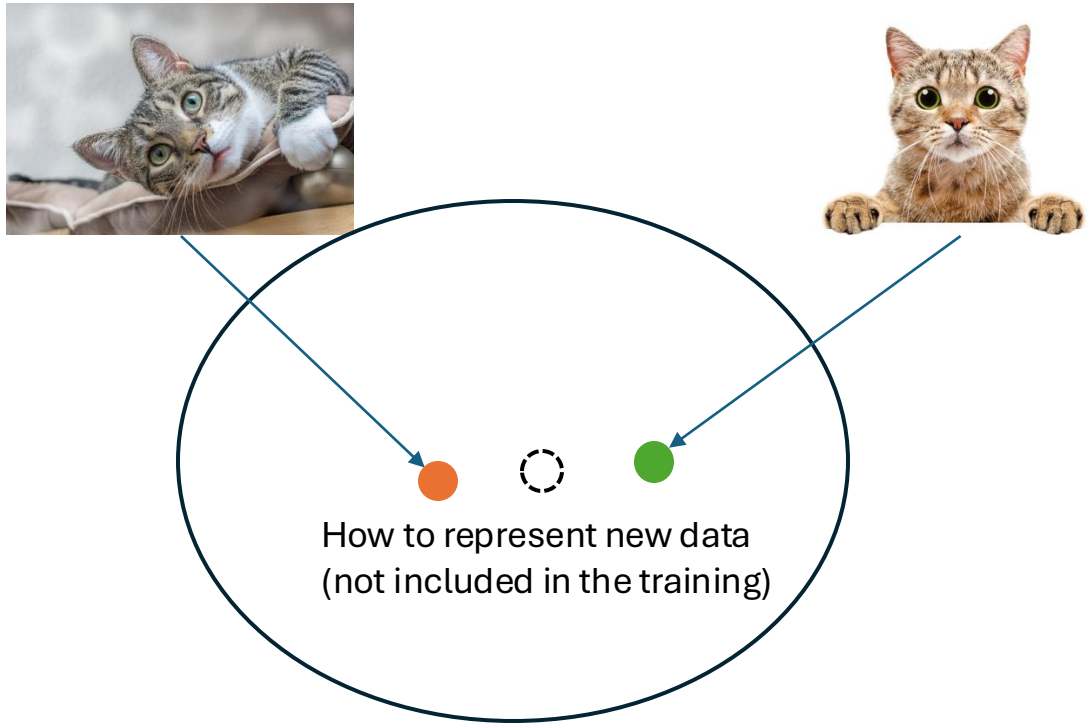




+ Easy to use, simple structure, fast to train
- Identity mapping is prone to overfit the data.
- non-interpolatable and non-smooth latent space
- Limited capacity of generating **new** data

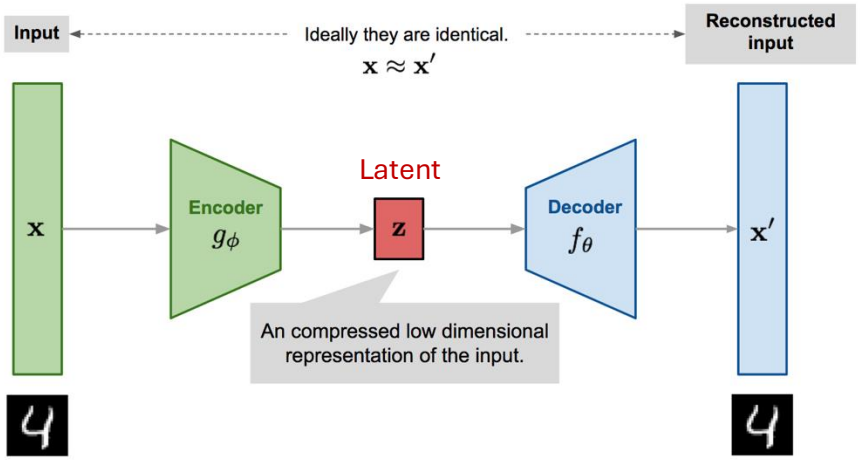
AutoEncoder (AE)

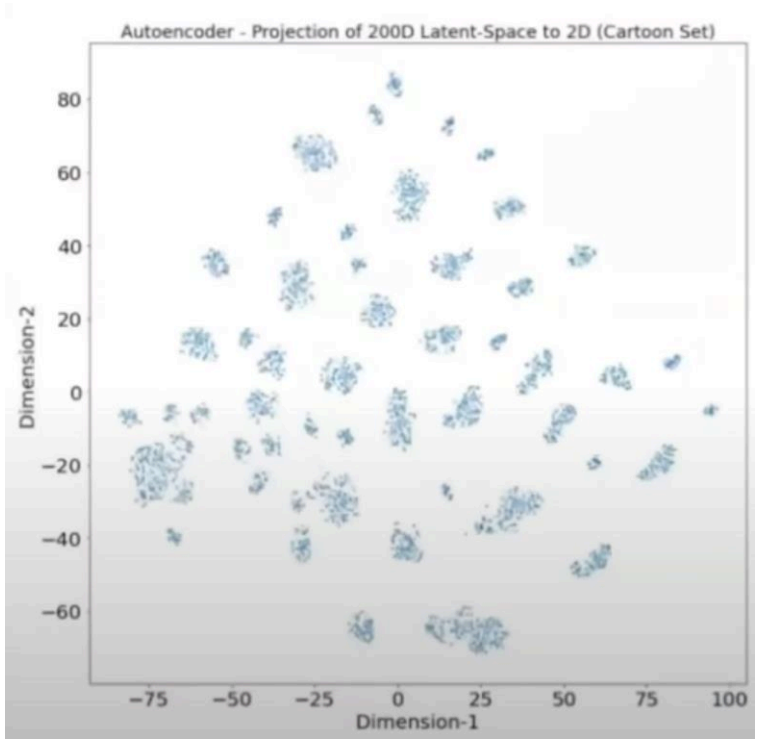




+ Easy to use, simple structure, fast to train
- Identity mapping is prone to overfit the data.
- non-interpolatable and non-smooth latent space
- Limited capacity of generating **new** data

AutoEncoder (AE)





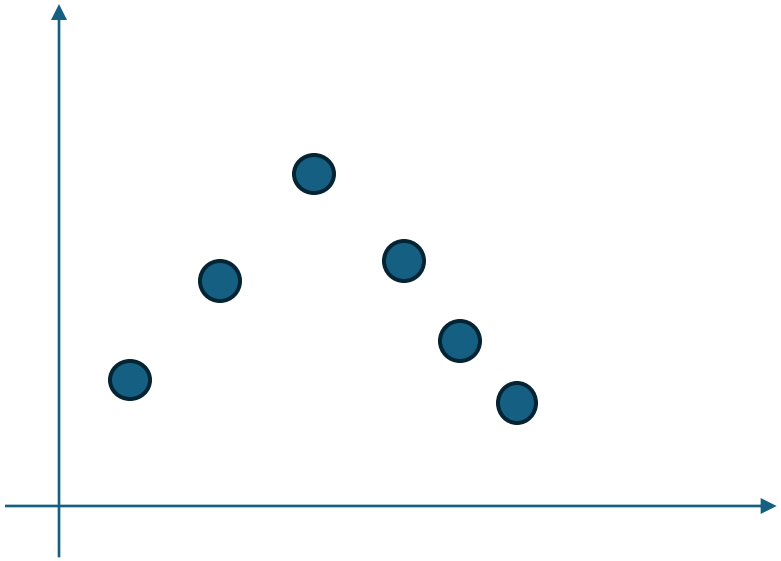
Latent Space

+ Easy to use, simple structure, fast to train
- Identity mapping is prone to overfit the data.
- non-interpolatable and non-smooth latent space
- Limited capacity of generating **new** data

Variational AutoEncoder

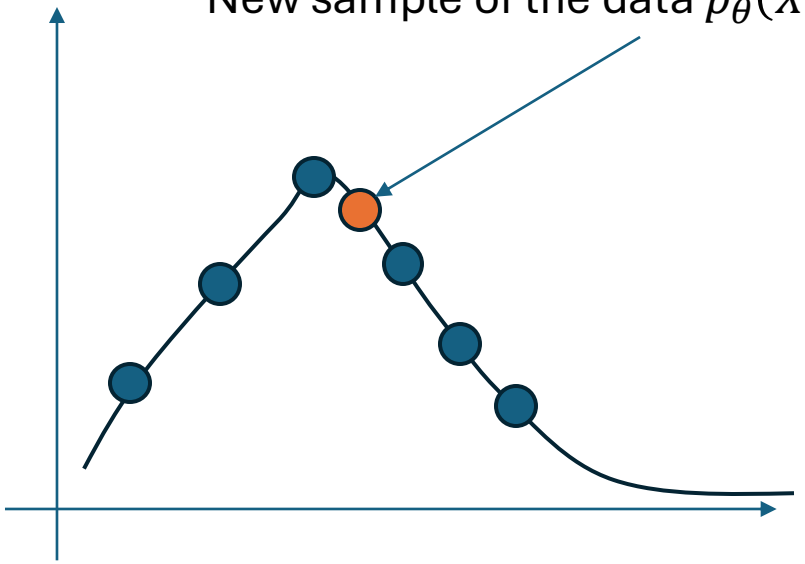
Variational AutoEncoder (VAE)

Instead of learning data one by one:
Solution: Learn a distribution with a hidden latent z!



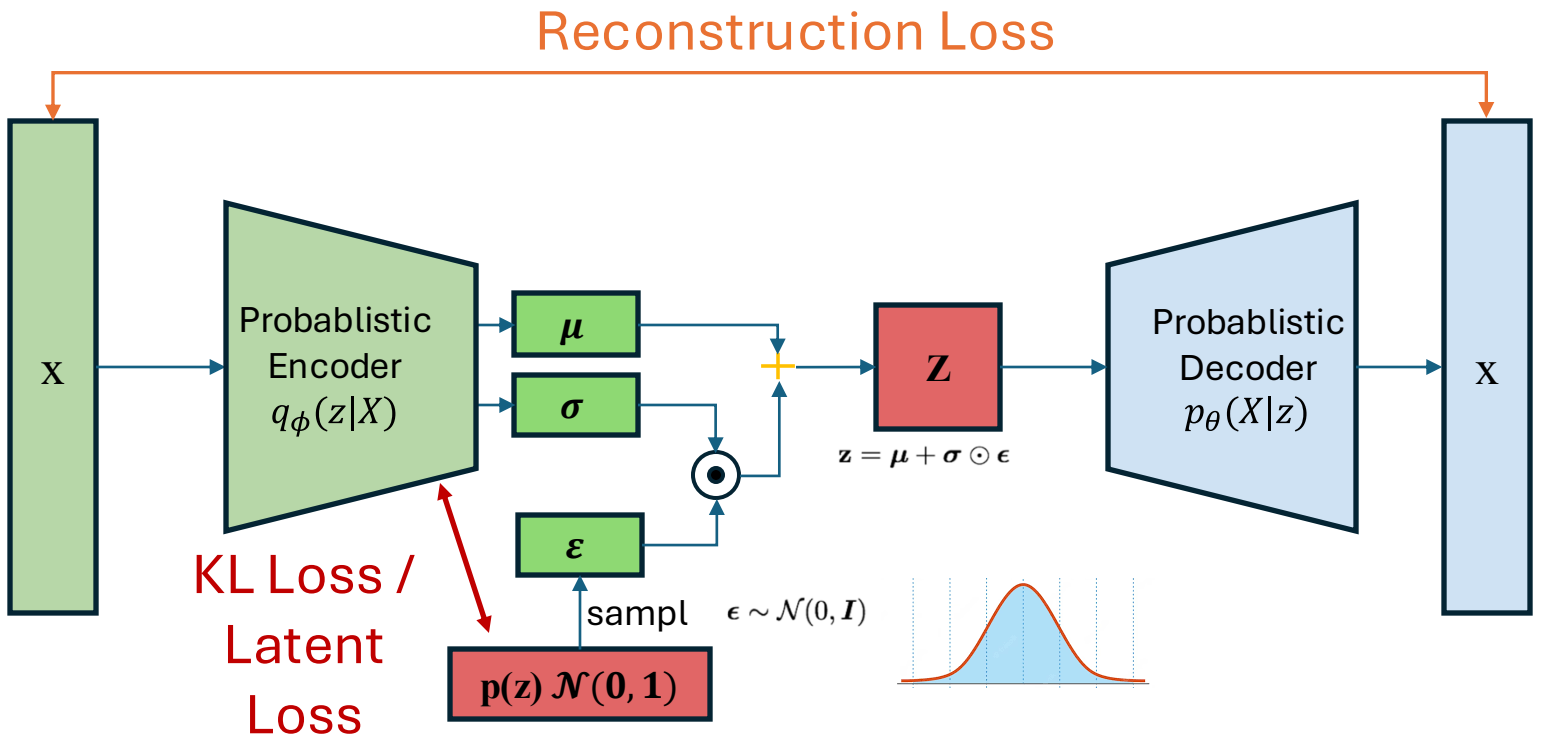
Data

New sample of the data $p_\theta(X|z); Z \sim p_\theta(z)$



Distribution of the Data with a hidden latent $p_\theta(X|z)p_\theta(z)$

Variational AutoEncoder (VAE)



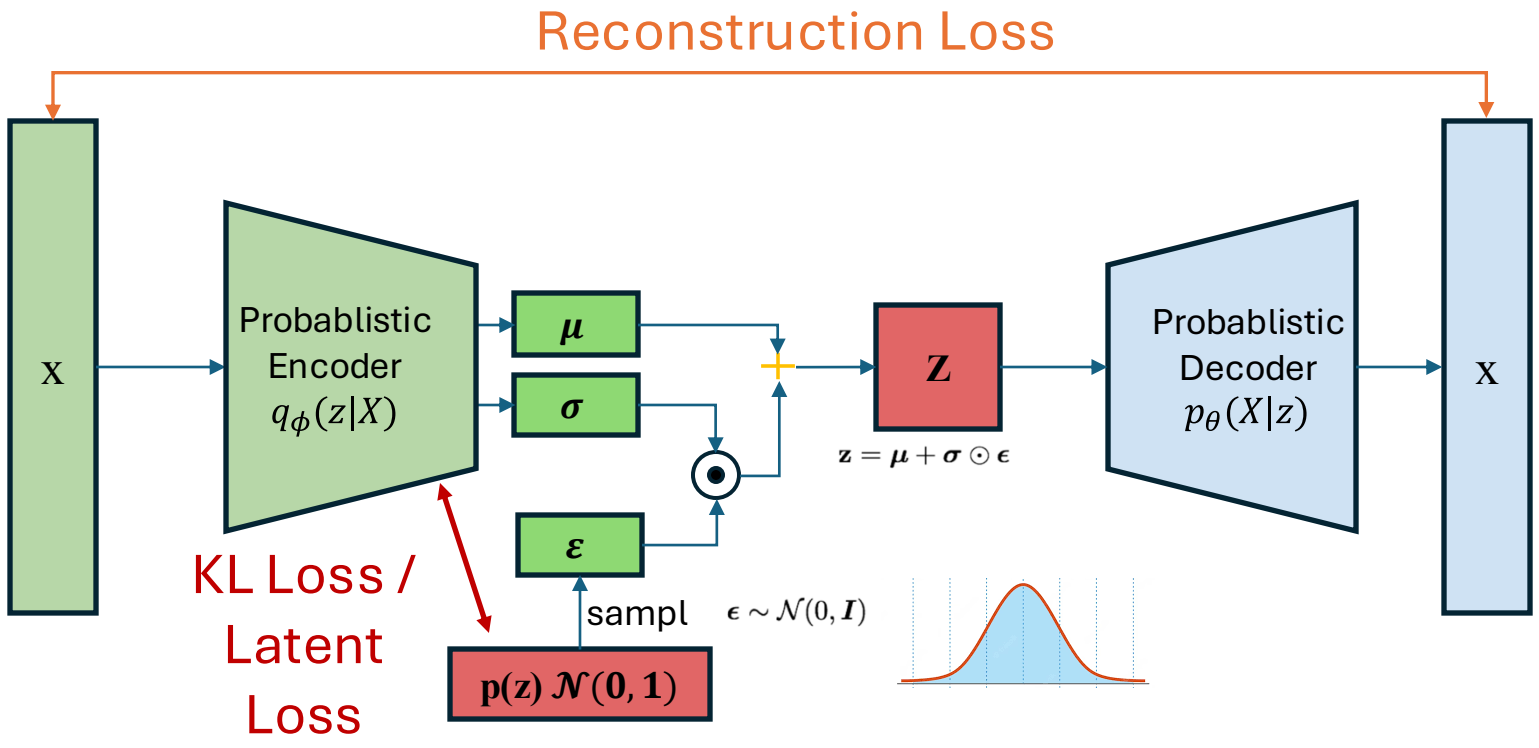
$$L_{\text{VAE}}(\theta, \phi) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

θ: Parameter of decoder

ϕ: Parameter of encoder

Kingma & Welling, 2014

Beta-VAE



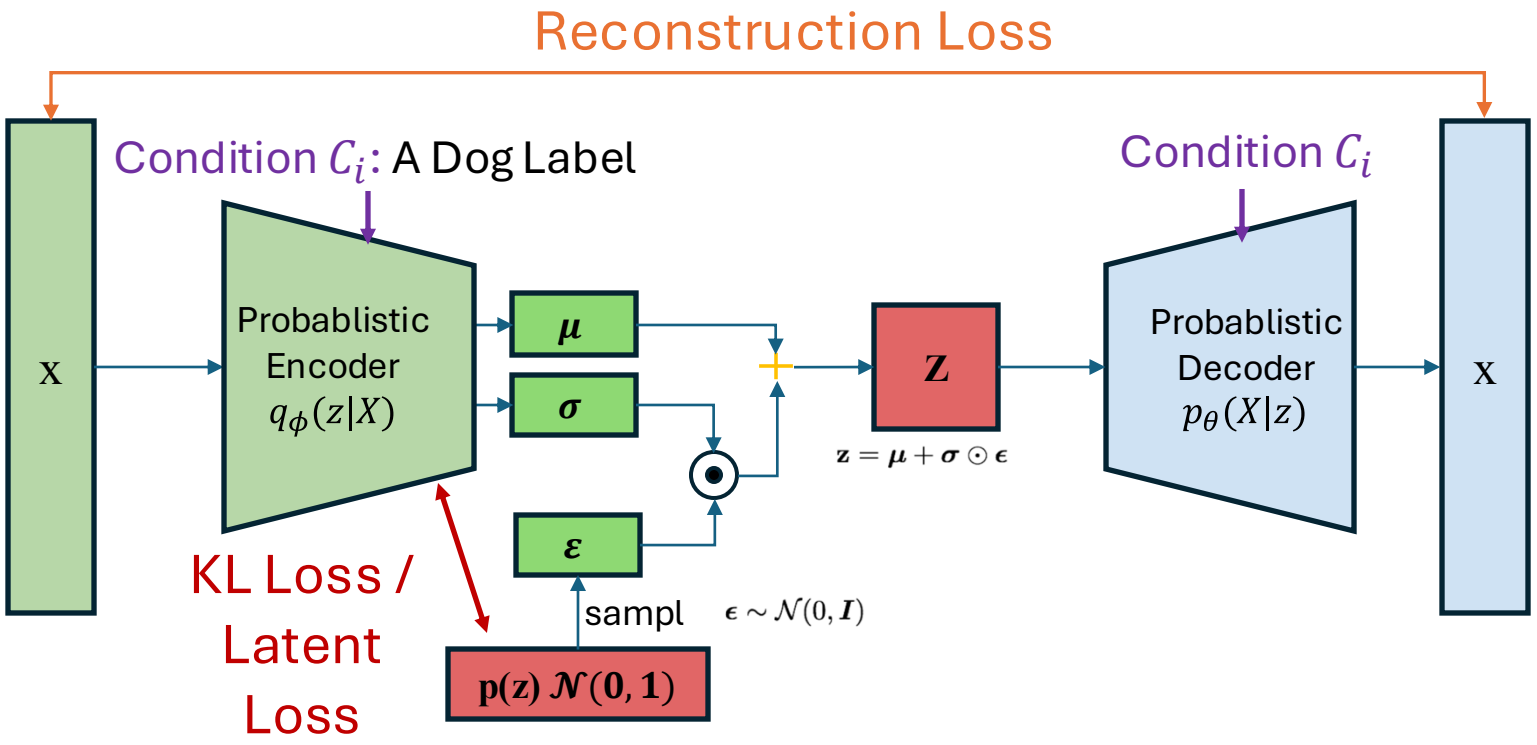
$$L_{\text{VAE}}(\theta, \phi) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

θ: Parameter of decoder

ϕ: Parameter of encoder

Kingma & Welling, 2014

Conditional VAE (CVAE)



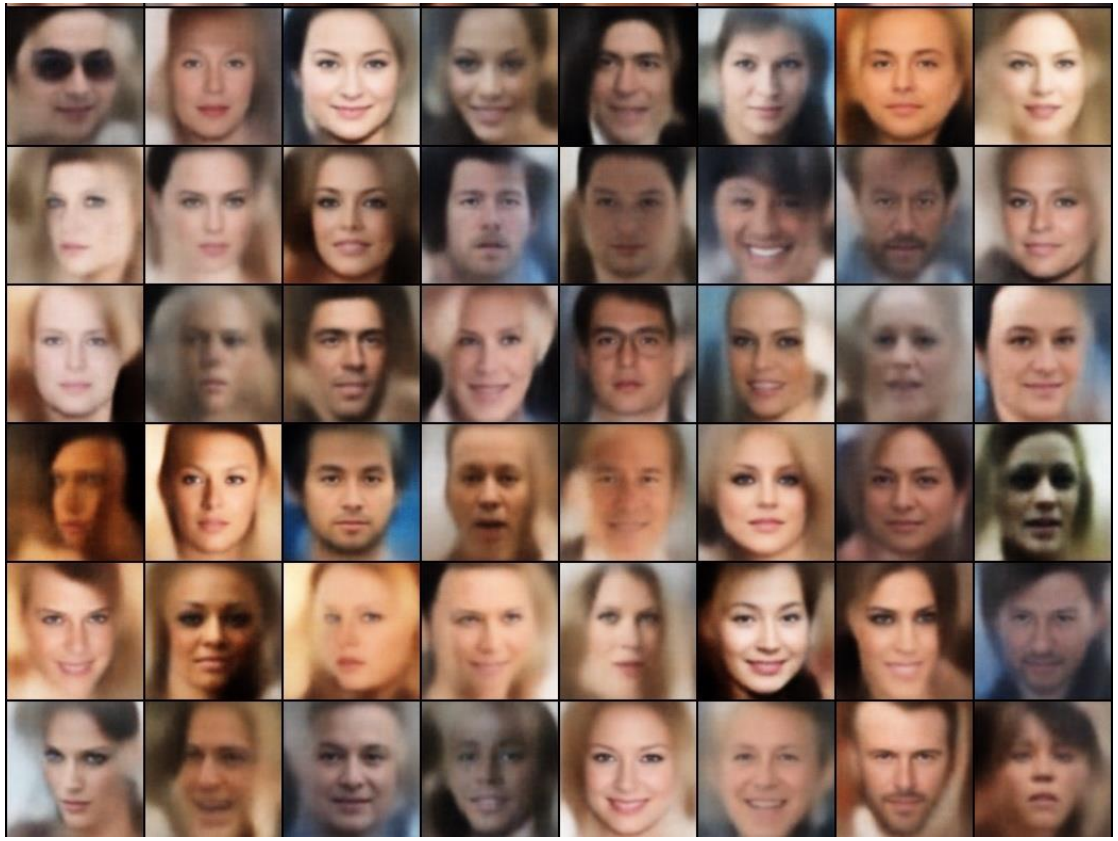
$$L_{\mathrm{CVAE}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c}_i)} \log p_\theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c}_i) + \beta D_{\mathrm{KL}} \left(q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}_i) \| p_\theta(\mathbf{z})\right)$$

Kingma & Welling, 2014

θ: Parameter of decoder

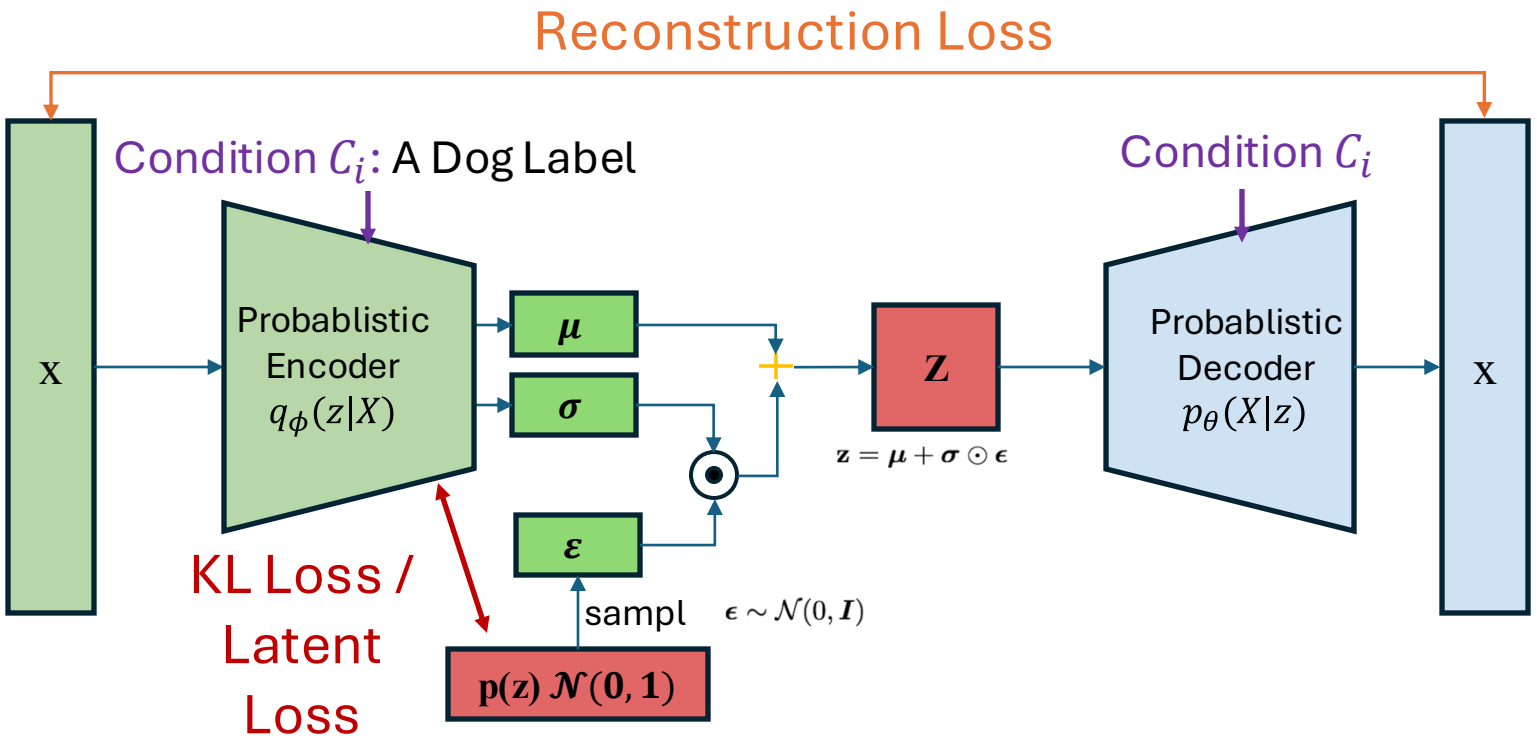
ϕ: Parameter of encoder

VAE Examples



Samples from Vanilla VAE (Kingma & Welling, 2014) on dataset CelebA

Summary VAE



$$L_{\mathrm{CVAE}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c}_i)} \log p_\theta(\mathbf{x} \mid \mathbf{z}, \mathbf{c}_i) + \beta D_{\mathrm{KL}} \left(q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{c}_i) \| p_\theta(\mathbf{z})\right)$$

Kingma & Welling, 2014

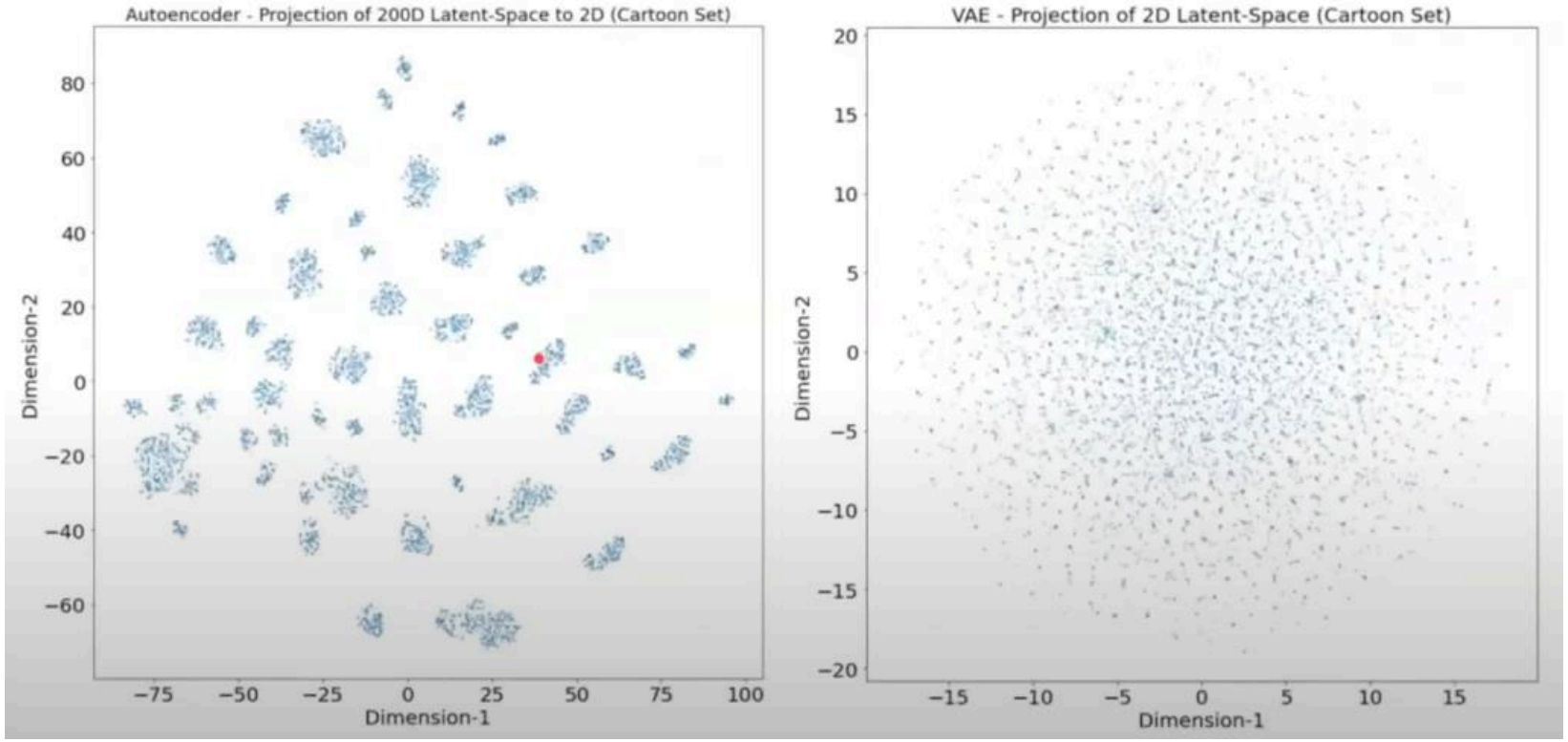
θ: Parameter of decoder

ϕ: Parameter of encoder

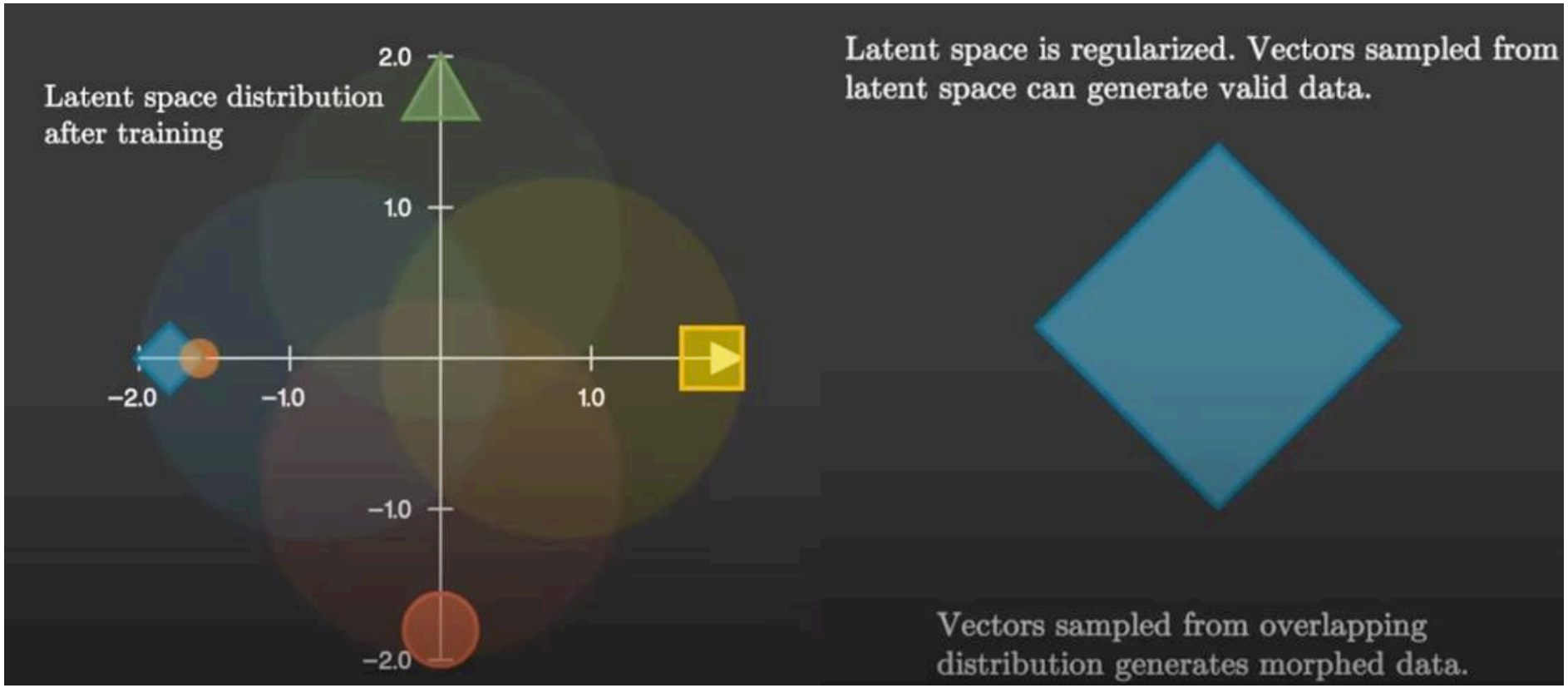
Problem with VAE

Guess?

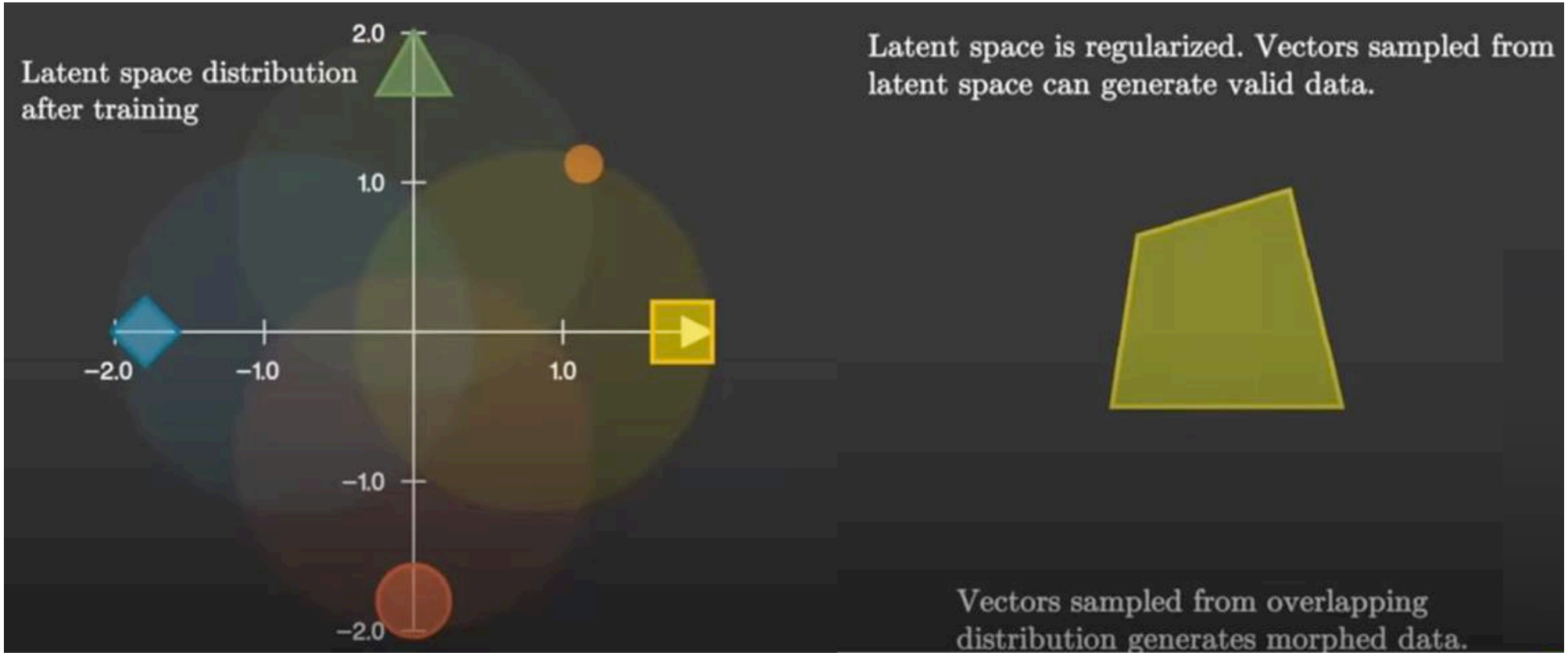
Problem with VAE



Problem with VAE

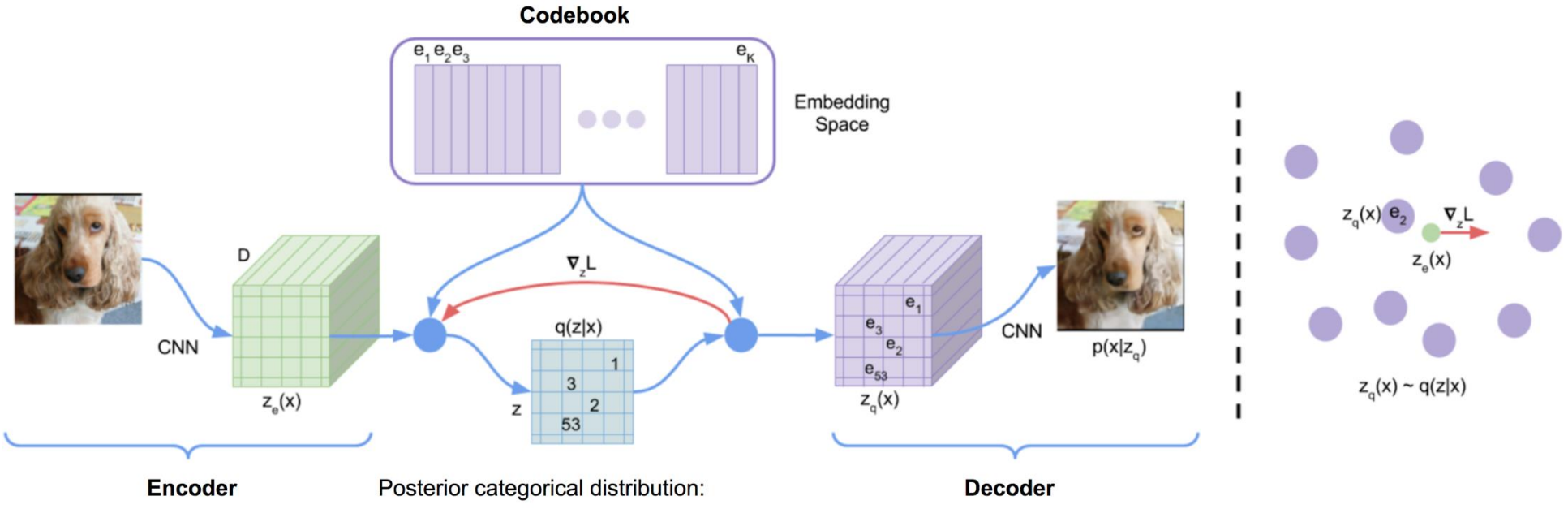


Problem with VAE



Vector Quantised-Variational AutoEncoder (VQVAE)

Training VQVAE



$$q(\mathbf{z} = \mathbf{e}_k|\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{reconstruction loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ loss}} + \underbrace{\beta\|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{commitment loss}}$$

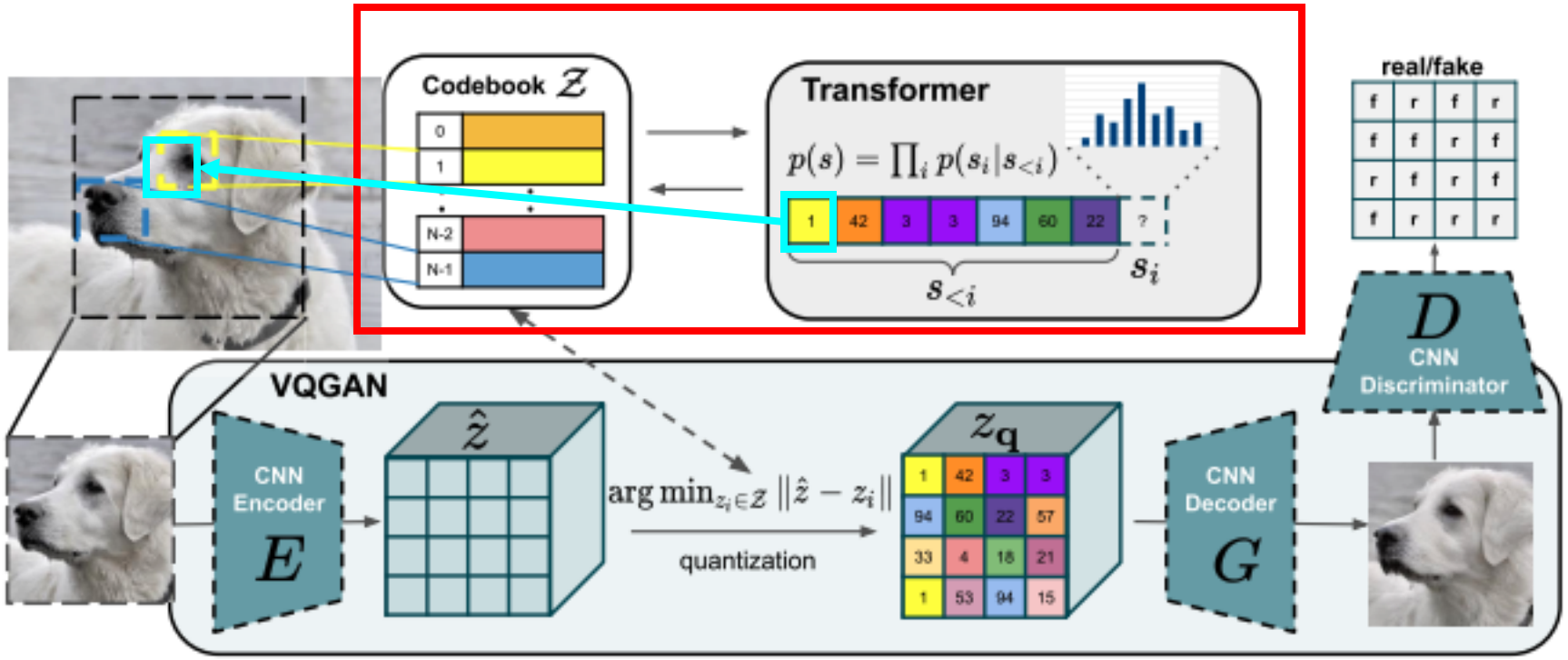
sg[.] for stop gradient

VQVAE

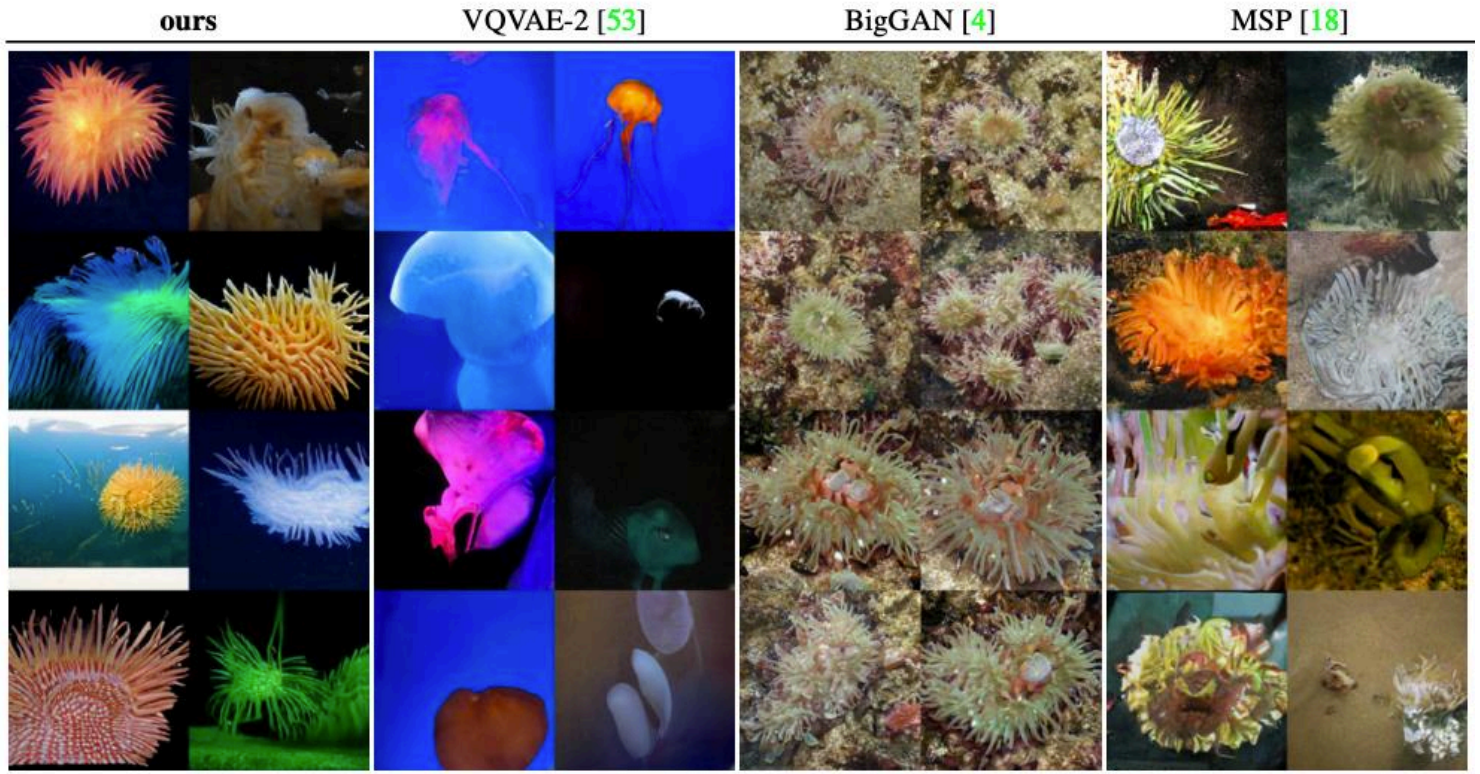
Sampled Results on ImageNet
VQVAE(Van den Oord, et al. 2017)

VQGAN

VQGAN


Codebook $\mathcal{Z}$
0
1
N-2
N-1
Transformer
$p(s) = \prod_i p(s_i|s_{<i})$
1 42 3 3 94 60 22 ?
$s_{<i}$
s_i
real/fake
f r f r
f f r f
r f r f
f r r r
D
CNN
Discriminator
VQGAN
CNN
Encoder
E
$\hat{z}$
$\arg\min_{z_i \in \mathcal{Z}} \|\hat{z} - z_i\|$
quantization
z_q
1 42 3 3
94 60 22 57
33 4 18 21
1 53 94 15
CNN
Decoder
G

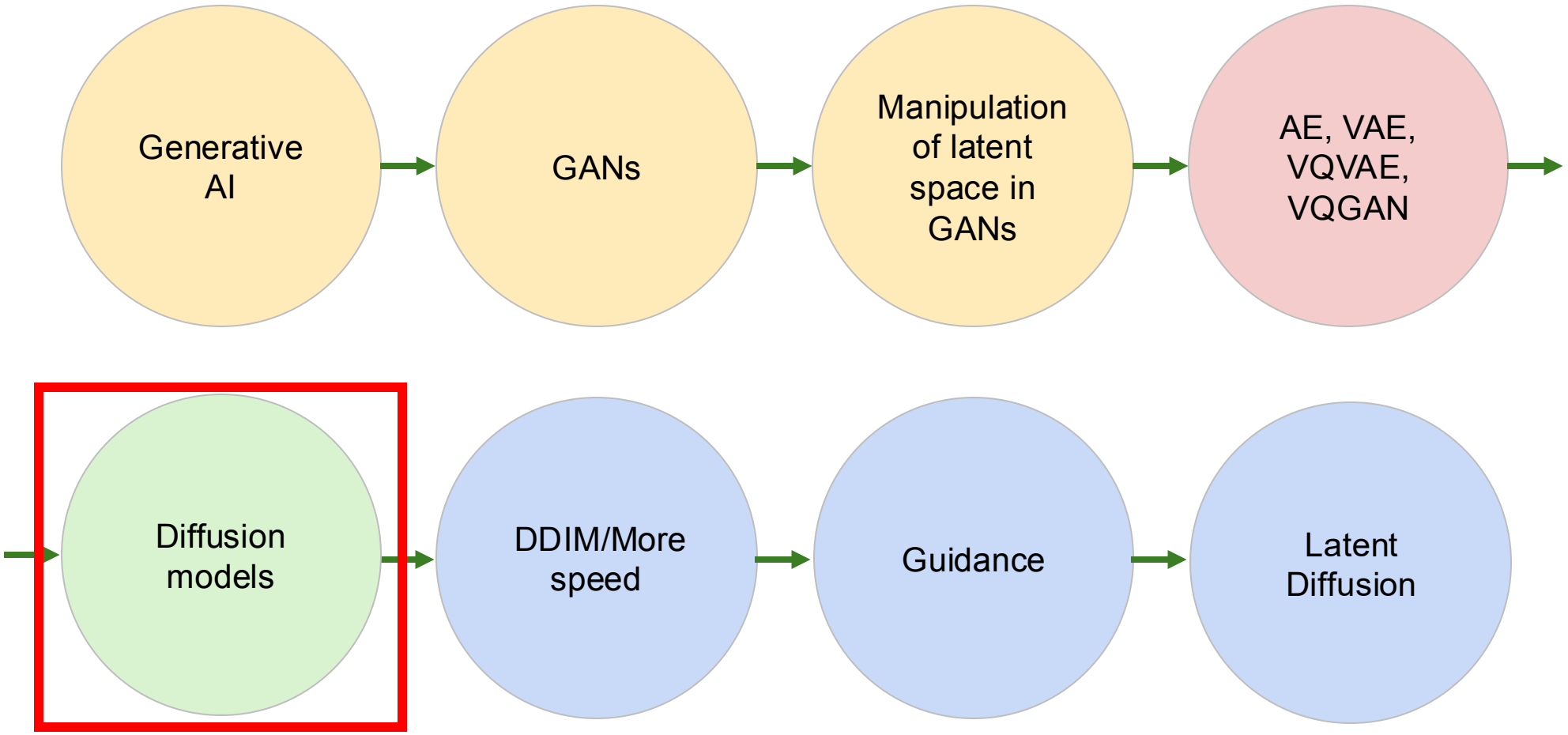
VQGAN



VQGAN(Esser et al. 2021) compared with VQVAE and other methods

Esser et al. 2021

Multimodal Generative AI with Diffusion

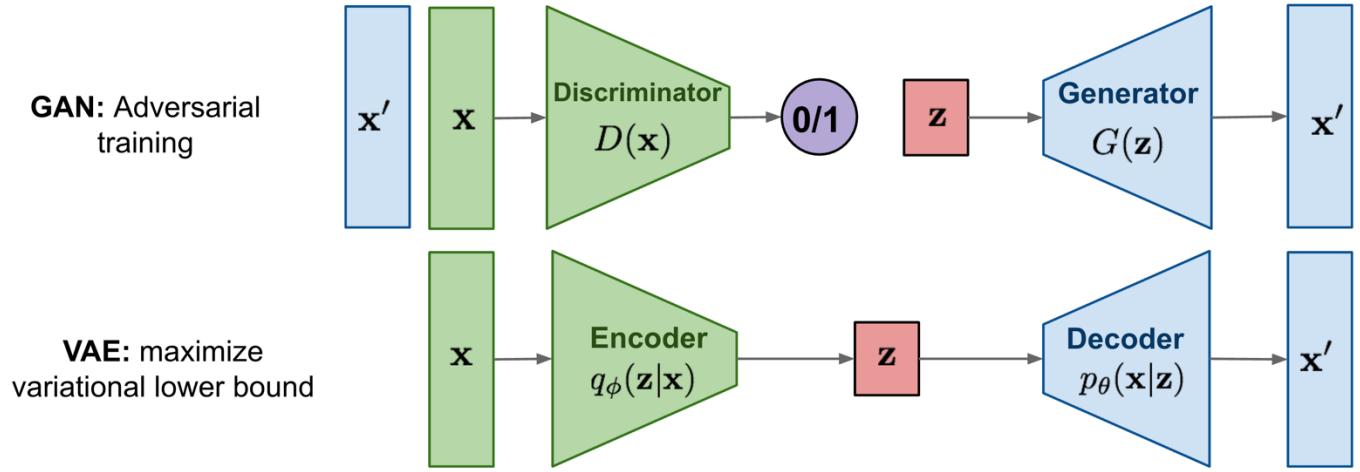


Part V: Outline

Diffusion models

- Learn distribution
- Forward process
- Reverse process
- Loss
- Network and Convergence
- Examples

Generative Models

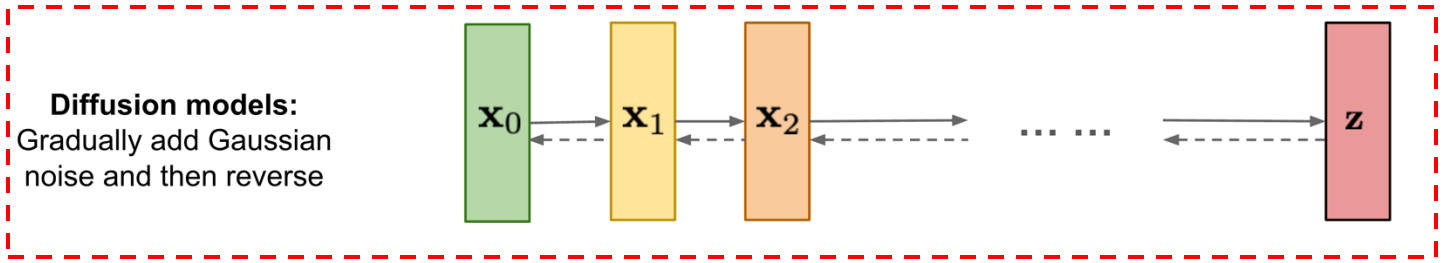


$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

unstable training
and mode collapse (learning data, instead of distribution)

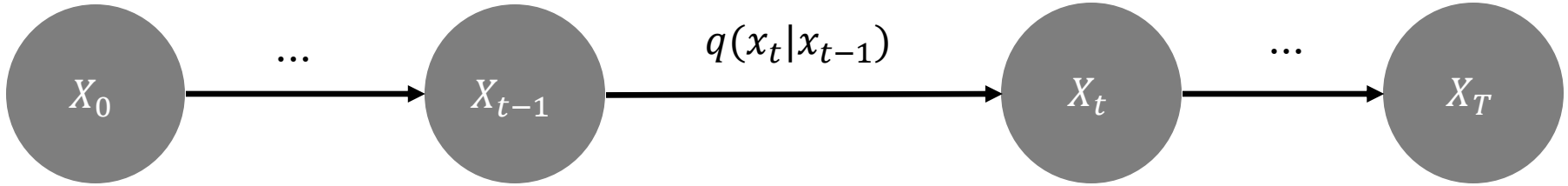
$$L_{\text{VAE}}(\theta,\phi) = -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$
$$= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$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

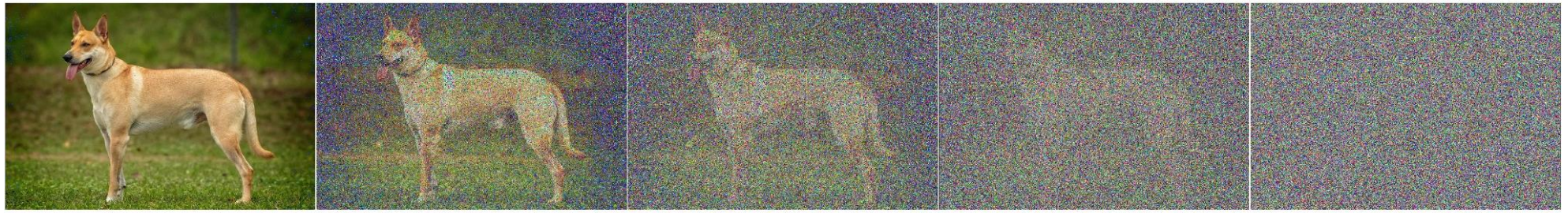
under-represtation of the distribution,
posteriori collapse (Gaussian Priori is not realistic)



Better representation capacity,
and learn the whole distribution.

Generative Objective: Forward Process



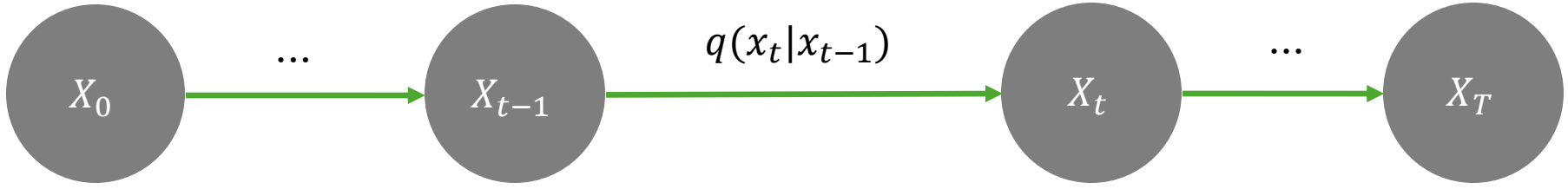


We call this **a Forward Process.**

- Original image at X_0 and pure noise at X_T
- We repeat the noising T times
- $\beta_t \in (0,1)$ is a noise schedule, ie. linear

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

DDPM: Forward Process



- Original image at X_0 and pure noise at X_T
- We repeat the noising T times
- $\beta_t \in (0,1)$ is a noise schedule

Forward:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

("Shortcut")
Sample any step using x_0 :

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

$$\alpha_t = 1 - \beta_t \qquad \bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$$

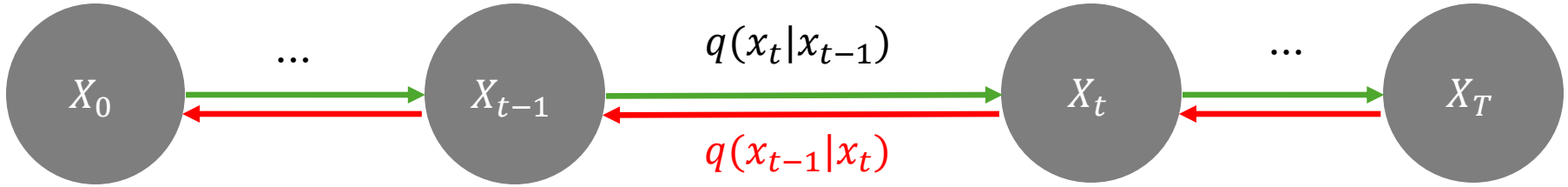
Reverse process

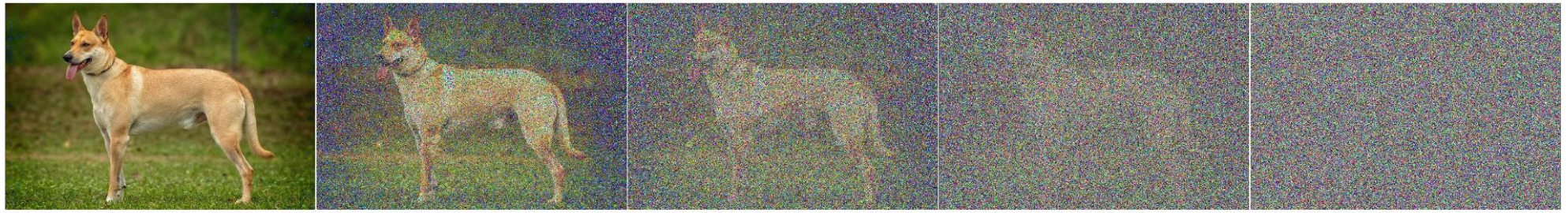
Generative Objective: Reverse Process *Summary*

1. In the “Reverse Diffusion process,” the idea is to reverse the forward diffusion process.
2. We slowly and iteratively try to reverse the corruption performed on images in the forward process.
3. The reverse process starts where the forward process ends.
4. The benefit of starting from a simple space is that we know how to get/sample a point from this simple distribution (think of it as any point outside the data subspace).
5. And our goal here is to figure out how to return to the data subspace.
6. However, the problem is that we can take infinite paths starting from a point in this “simple” space, but only a fraction of them will take us to the “data” subspace.
7. In diffusion probabilistic models, this is done by referring to the small iterative steps taken during the forward diffusion process.
8. The PDF that satisfies the corrupted images in the forward process differs slightly at each step.
9. Hence, in the reverse process, we use a deep-learning model at each step to predict the PDF parameters of the forward process.
10. And once we train the model, we can start from any point in the simple space and use the model to iteratively take steps to lead us back to the data subspace.
11. In reverse diffusion, we iteratively perform the ***“denoising”*** in small steps, starting from a noisy image.
12. This approach for training and generating new samples is much more stable than GANs and better than previous approaches like variational autoencoders (VAE) and normalizing flows.

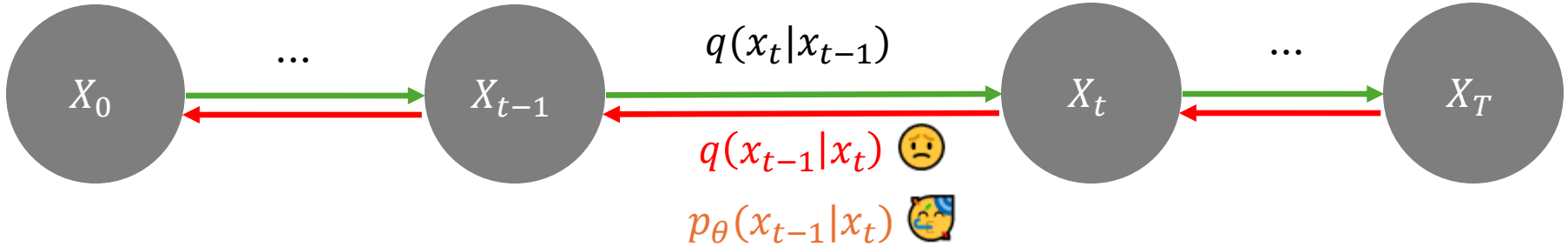
[Vaibhav Singh]

Generative Objective: Reverse Process





Generative Objective: Reverse Process



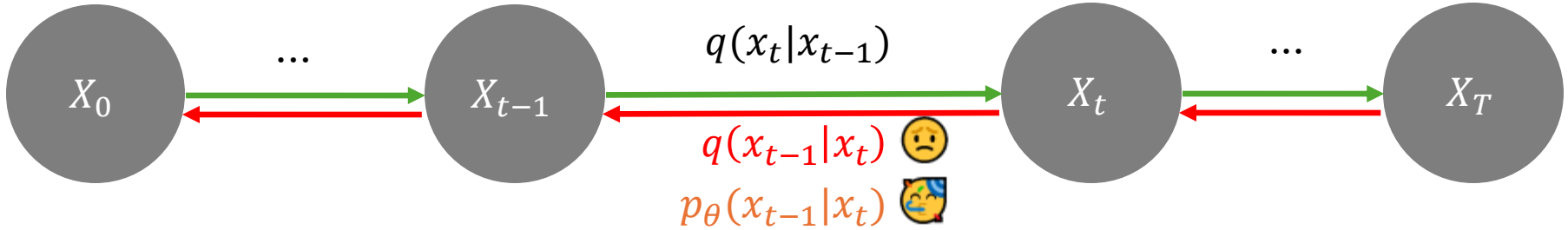
A very nice property of Gaussian:
if $q(x_t|x_{t-1})$ is a Gaussian with small β (another reason we need many steps!)
→ then, $q(x_{t-1}|x_t)$ is also a Gaussian.
Therefore, we learn this Gaussian's mean and variance by a network approximated $p_\theta(x_{t-1}|x_t)$

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

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

Learnable parameters

DDPM: Reverse (=Generative) Process



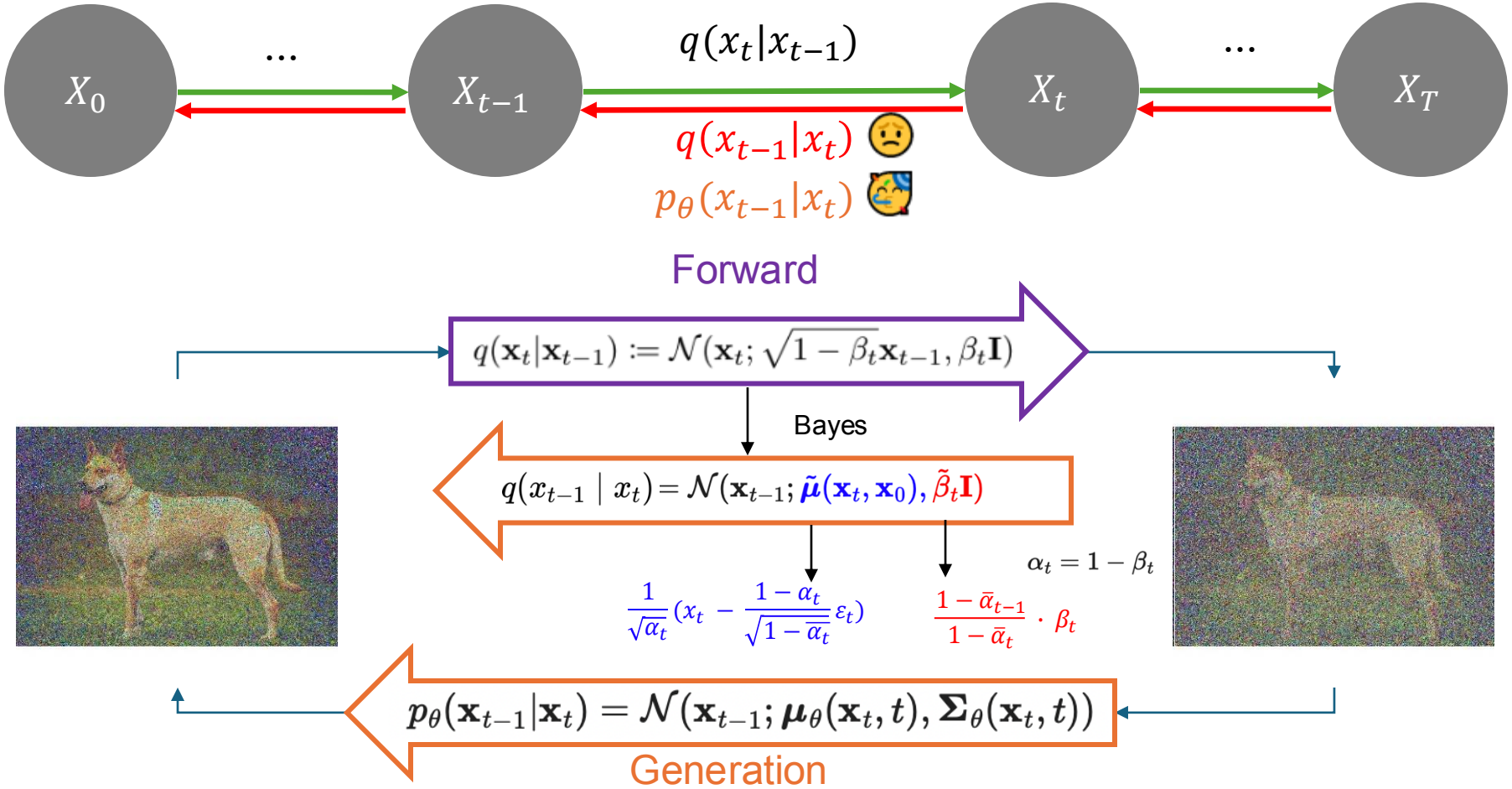
Generation:

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

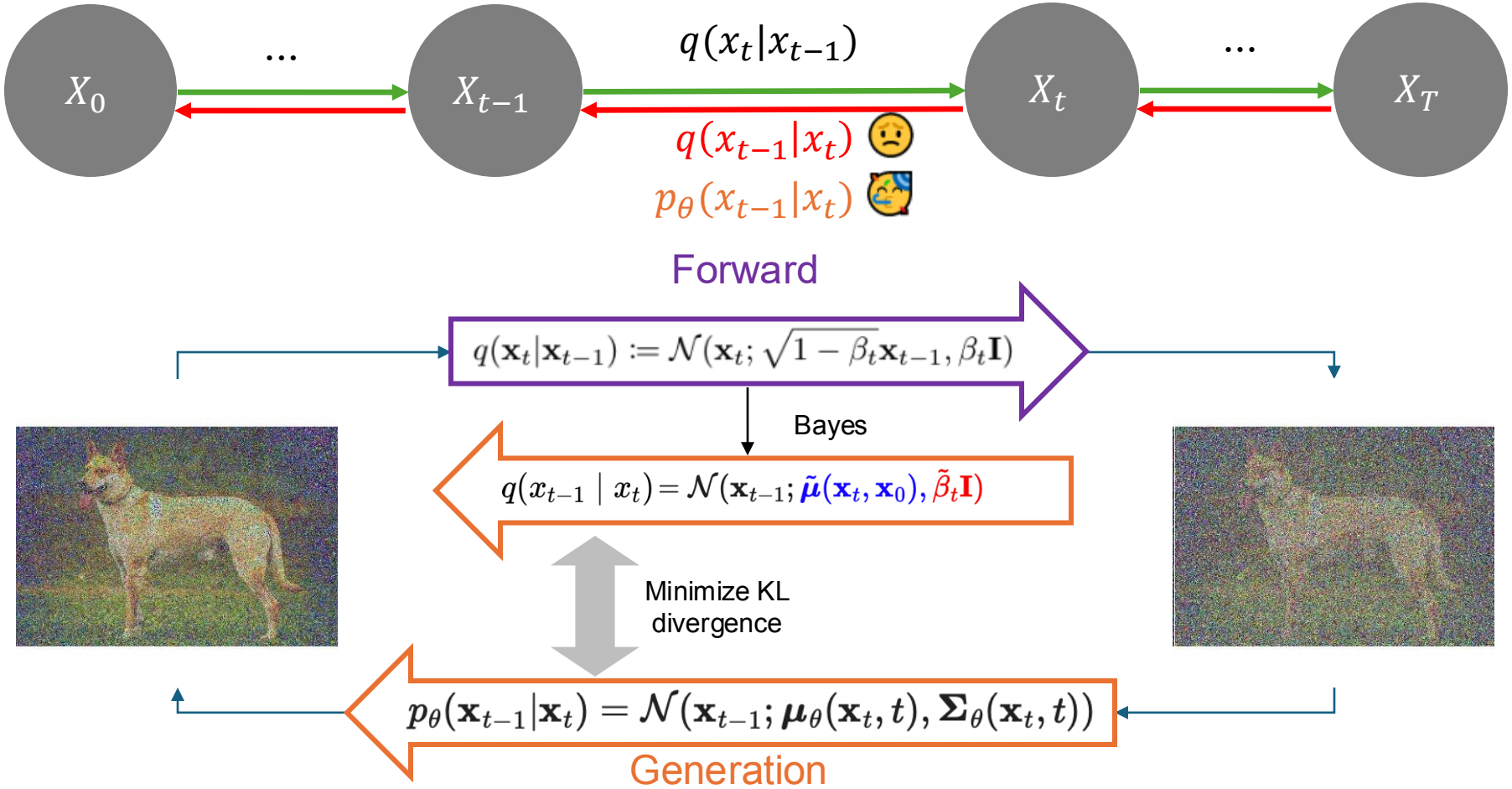
Learnable parameters

A very nice property of Gaussian:
if $q(x_t|x_{t-1})$ is a Gaussian with small β (another reason we need many steps!)
→ then, $q(x_{t-1}|x_t)$ is also a Gaussian.
Therefore, we learn this Gaussian's mean and variance by a network approximated $p_\theta(x_{t-1}|x_t)$

DDPM: Reverse/Generative Process

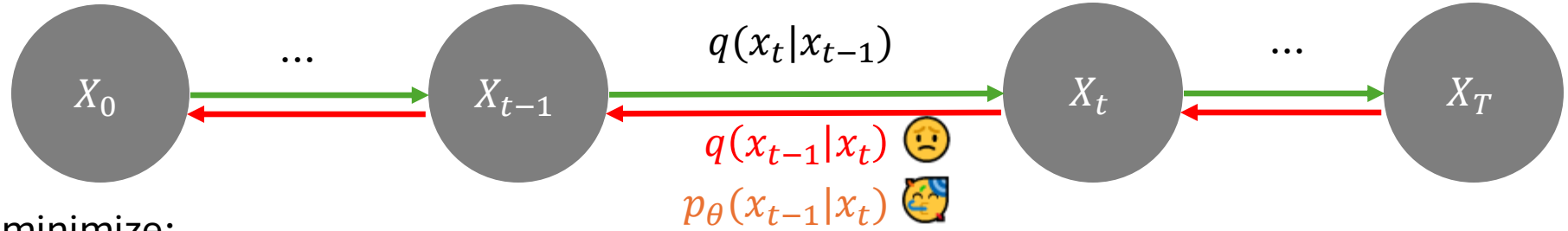


DDPM: Reverse/Generative Process



Loss

Generative Objective: Loss



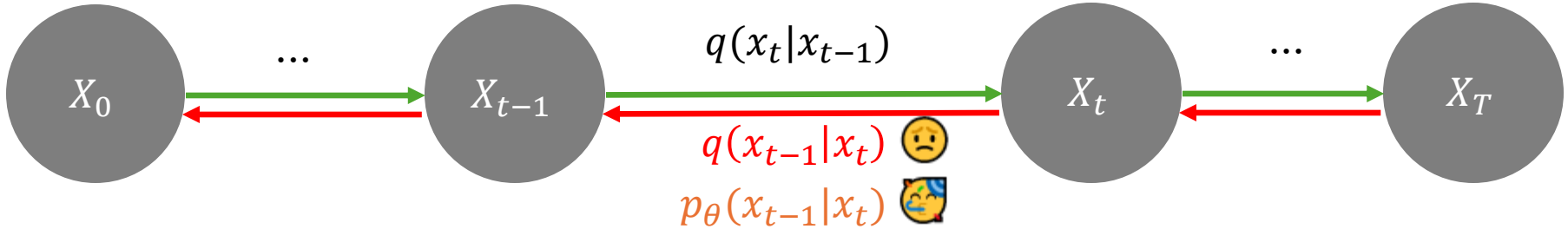
We want to minimize:

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$

$$\text{where } L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$

$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1$$

Generative Objective: Loss



We want to minimize:

$$L_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0$$
$$\text{where } L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))$$
$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \le t \le T-1$$

=

Minimizing predicted noise based on data:

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right]$$

In code:

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

Algorithm 2 Sampling

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

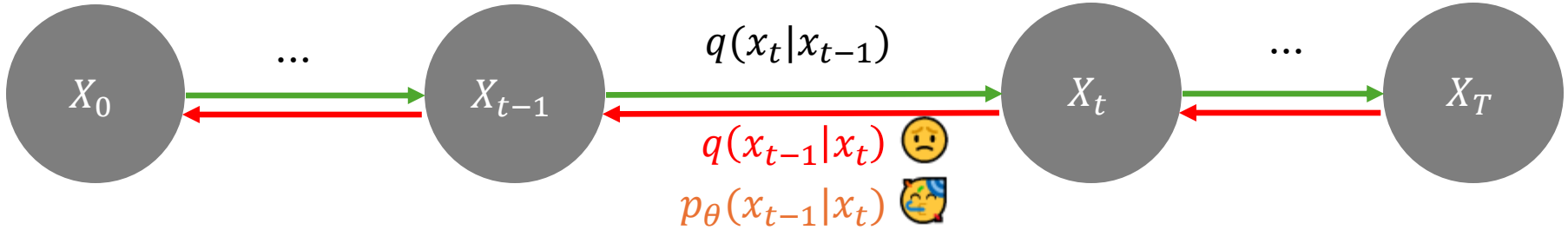
$x \sim N(\mu, \sigma^2)$
$z \sim N(0,1)$
$x = \mu + \sigma z$

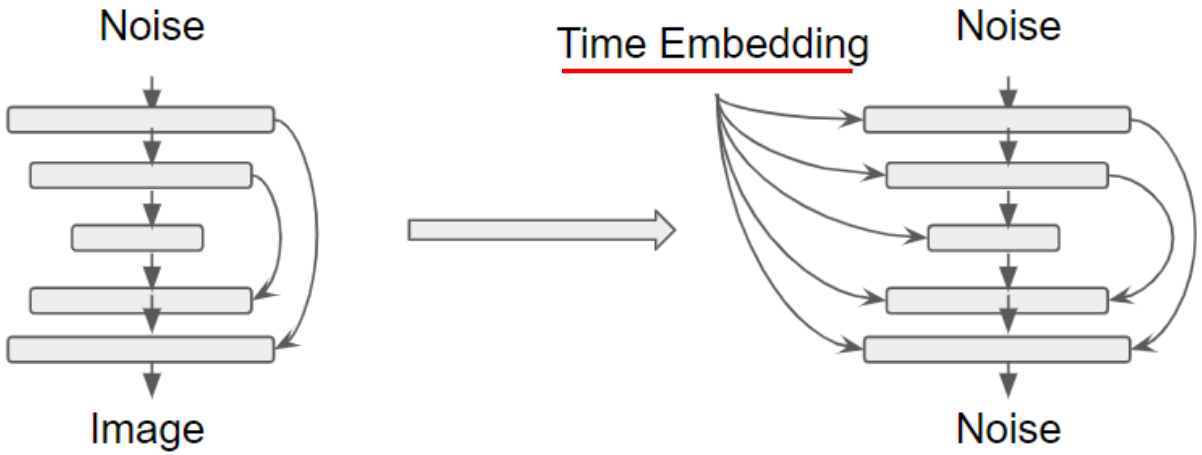
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$$\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta\right) \qquad \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

Network/Convergence/Examples

Generative Objective: Which networks?



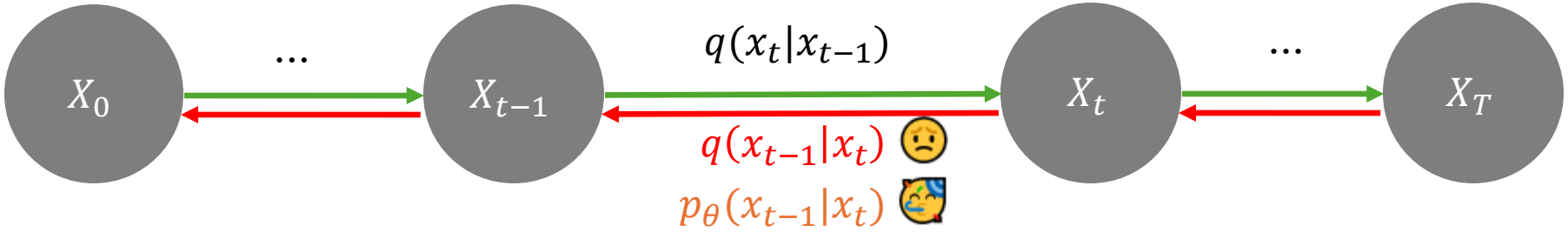


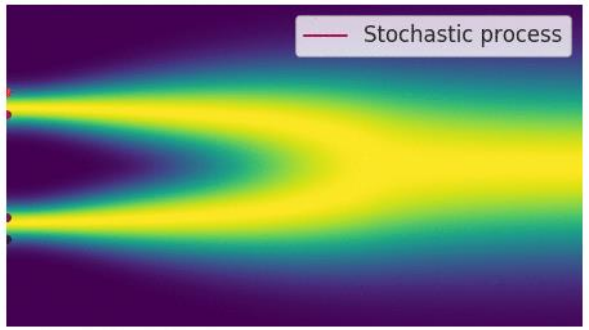
A standard U-Net to predict noise from previous noise and time information.

U-Net for standard encoding

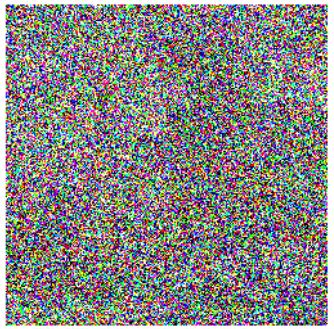
A (very simple) U-Net for diffusion Model

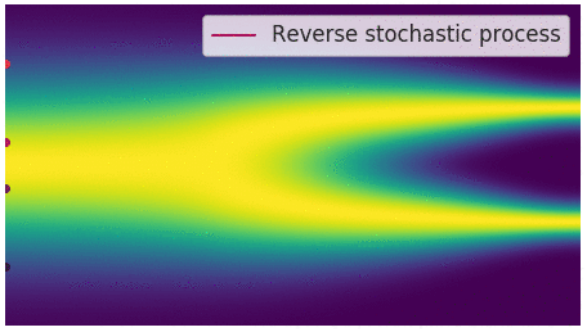
Generative Objective: Reverse Process





Stochastic noising





Stochastic denoising

Generative Objective: Reverse Process
Convergence

- GANs: if the Discriminator can successfully differentiate between real/fake then we stop the training

- VAE: reconstruction loss (meaningful)

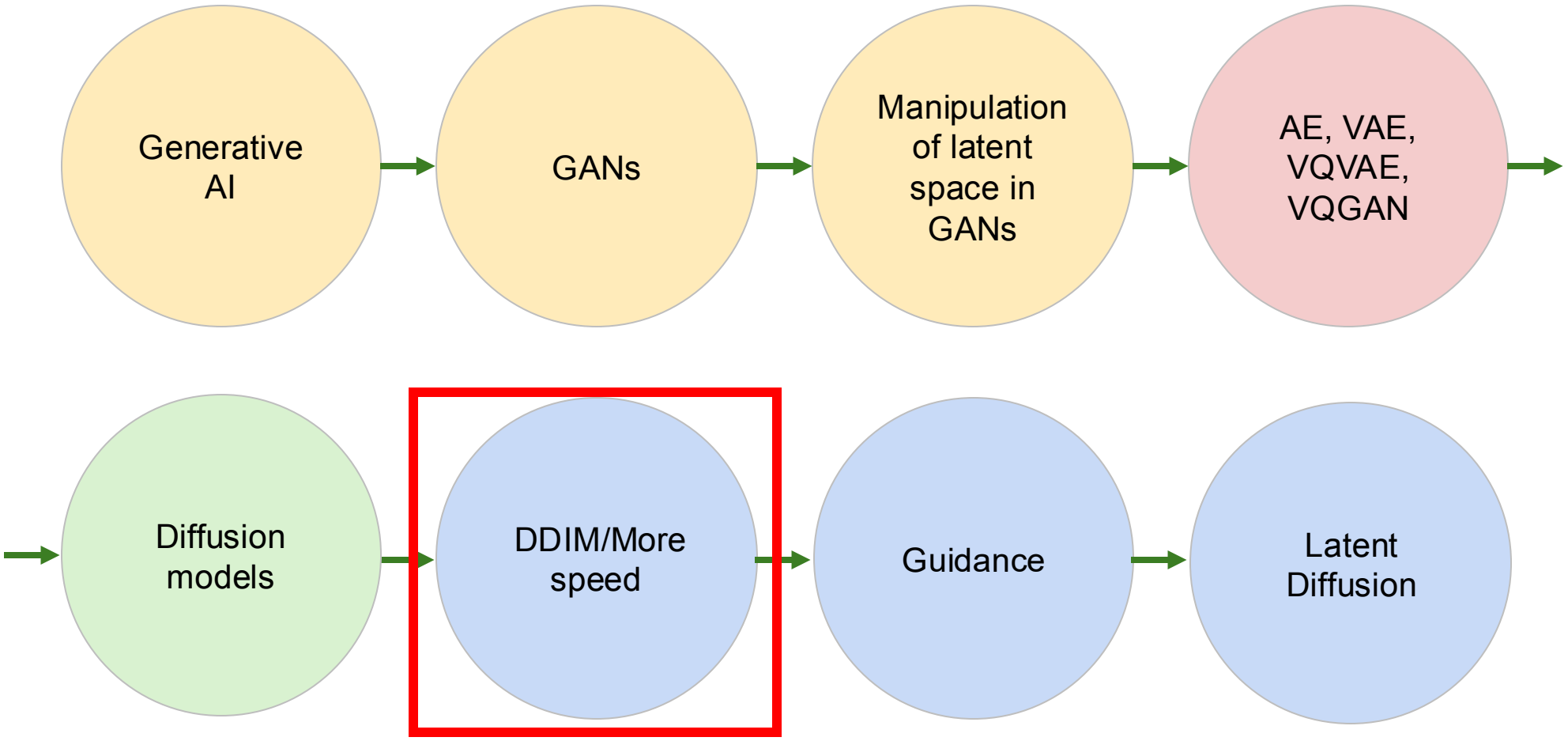
- Diffusion models: complicated: we need to measure the distance between two distributions → FID

Results: Denoising Diffusion Probabilistic Model (DDPM)

Sampled results:
LSUN Church Dataset

Sampled results:
LSUN Church Bedroom

Multimodal Generative AI with Diffusion



Problem with DDPM

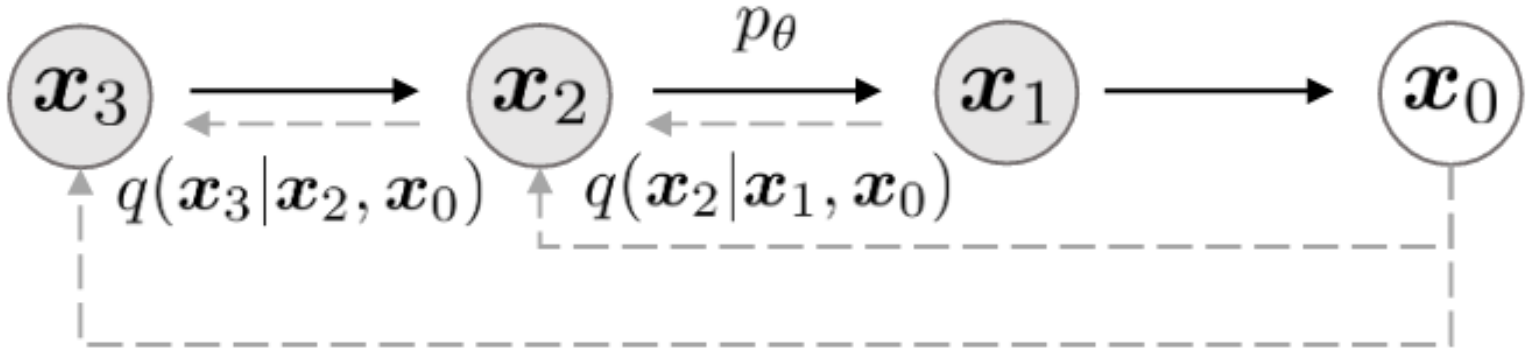
- Problem with DDPM: many iterations to produce high quality sample
  - Generative process approximates the reverse of the forward process, which could have thousands of steps; iterating over all the steps is required to produce a single sample
- Slower than GANs (one pass through the network)
- E.g. 50k images 23x23
  - DDPM: 20h
  - GAN: <1 min
- E.g. 50k images 256x256
  - DDPM: 1000h!

→

- Denoising Diffusion Implicit Models (DDIM)
  - Same objective function as DDPM
  - Faster sampling

Generalization of DDPMs

- A non-Markovian forward process



$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}\ \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\cdot\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \tilde{\beta}_t\mathbf{I}\right) \qquad \tilde{\beta}_t = \sigma_t^2$$

Generalization of DDPMs

- Keeps some properties of the original process

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

$$\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$$

$$\alpha_t = 1 - \beta_t$$

- Distribution of xt is defined given xt-1 and x0: the process is no longer Markovian

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)} \qquad \tilde{\beta}_t = \sigma_t^2$$

Generalization of DDPMs
Stochasticity

- Sigma/beta modulate the stochasticity of the process $\tilde{\beta}_t = \sigma_t^2$

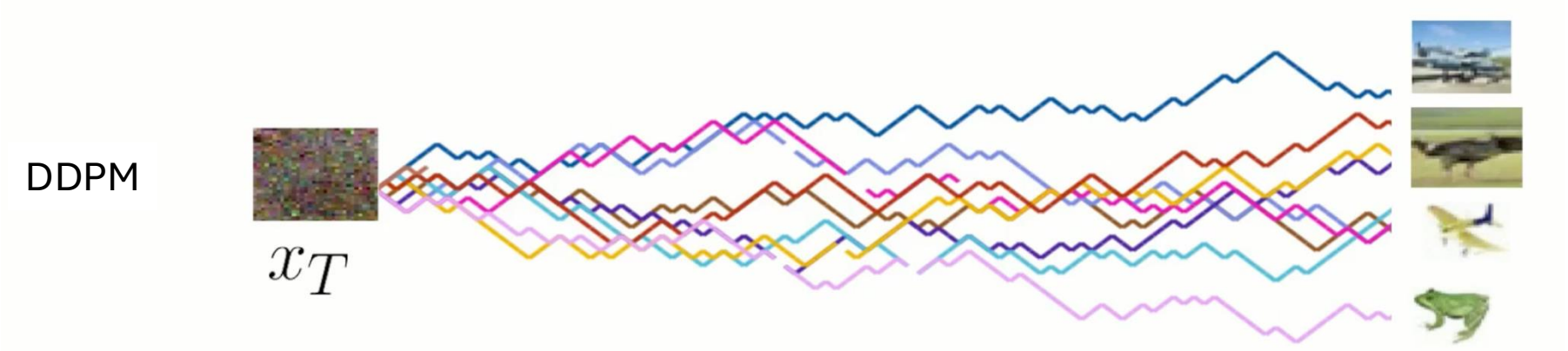
when $\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$ DDIM is DDPM

when $\tilde{\beta}_t(\eta) = \eta \frac{(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)} \cdot \beta_t$ interpolation between DDIM and DDPM, i.e. we control the sampling variance

when $\tilde{\beta}_t$ is zero, the DDIM is *deterministic* process
→ the same noise XT always leads to the same image x0

Generalization of DDPMs
Deterministic generation

- When Sigma/beta !=0: stochastic process



- When sigma/beta = 0 (DDIM): deterministic
  → the same noise XT always leads to the same image x0

Generalization of DDPMs
Training objective

- Training objectives are equivalent for any of the values of sigma/beta

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t\sim q(\boldsymbol{x}_0,\boldsymbol{x}_t)}[D_{\mathrm{KL}}(q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0))\|p_\theta^{(t)}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))]$$

- A model trained for the original DDPM process can be used for any process of the family

Generalization of DDPMs: Sampling

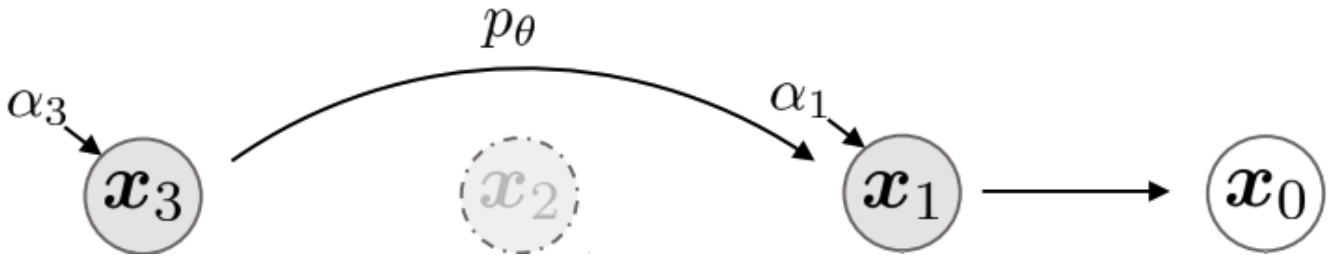
- Sampling becomes

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right).$$

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}}$$

Sampling acceleration

Generalization of DDPMs: Recap



$$p_\theta(x_{0:3}) := p_\theta(x_3) \cdot [p_\theta(x_1|x_3) \cdot p_\theta(x_0|x_1)] \times p_\theta(x_0|x_2)$$

A process defined on a subset $\{x_{\tau_1}, \ldots, x_{\tau_S}\}$

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{i=1}^{S} p_\theta^{(\tau_i)}(\boldsymbol{x}_{\tau_{i-1}}|\boldsymbol{x}_{\tau_i}) \times \prod_{t \in \bar{\tau}} p_\theta^{(t)}(\boldsymbol{x}_0|\boldsymbol{x}_t)$$

Experiments

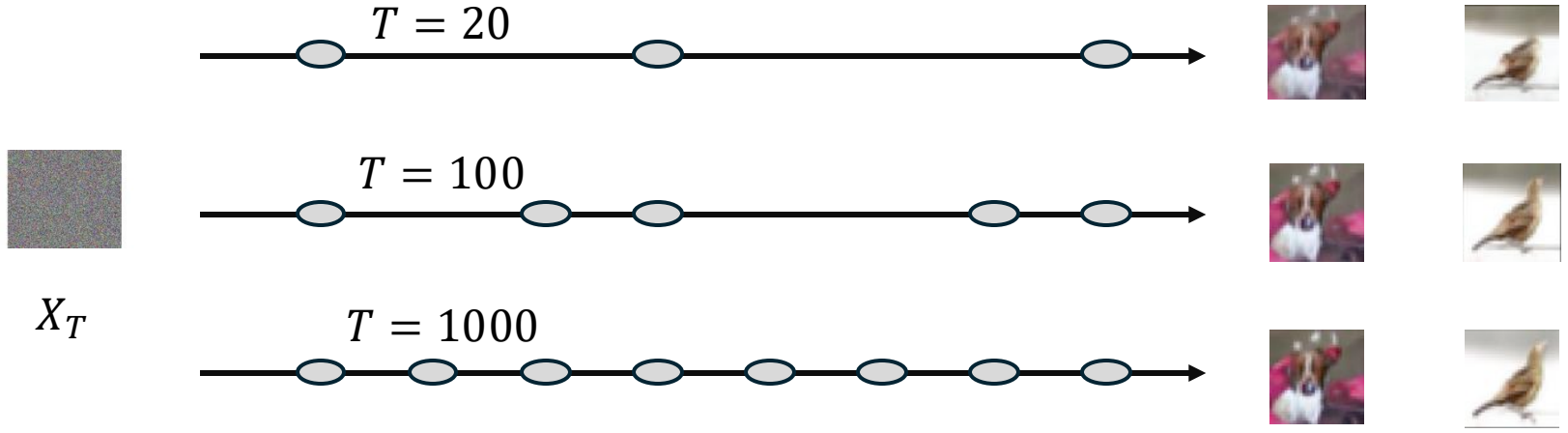
Generalization of DDPMs: Experiments

	S	CIFAR10 (32 × 32) 10	20	50	100	1000	CelebA (64 × 64) 10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
	$\hat{\sigma}$	367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

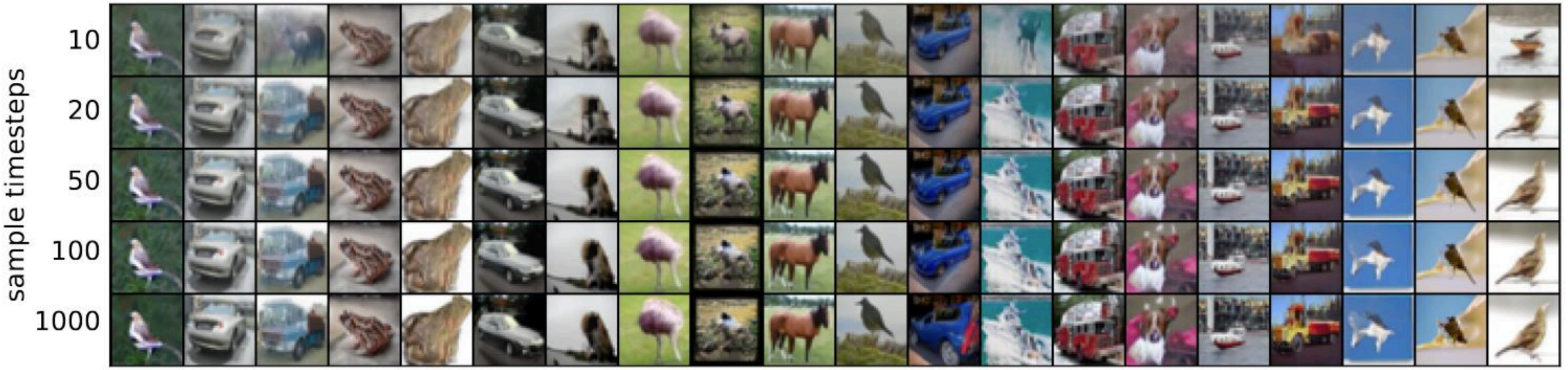
1. Sampling quality of DDIM better than DDPM for the same number of steps
2. For a given number of steps, when lowering the stochasticity, the sampling quality improves
3. DDIM reaches the same image quality of DDPM in 50 steps instead of 1000 steps → 20x speed up for the same image quality!

$$\tilde{\beta}_t(\eta) = \eta \frac{(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)} \cdot \beta_t$$

Generalization of DDPMs: Experiments

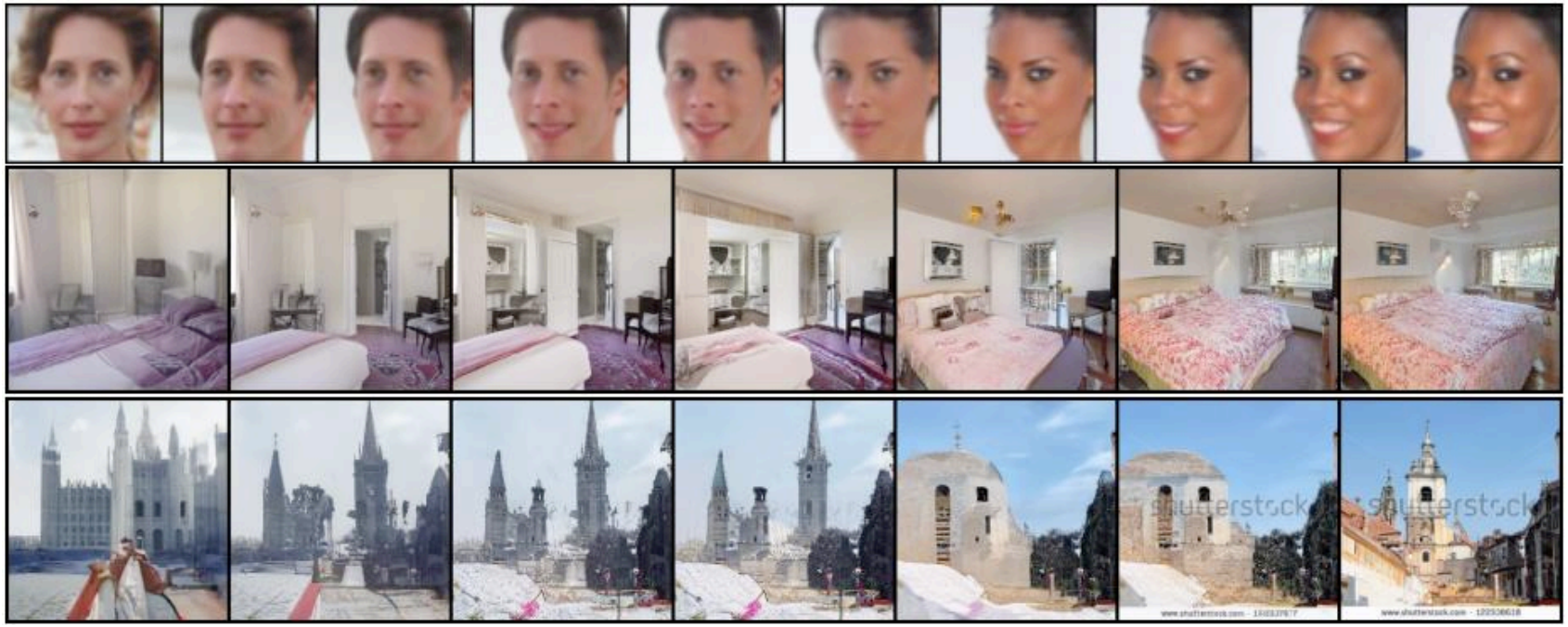


Generalization of DDPMs: Experiments

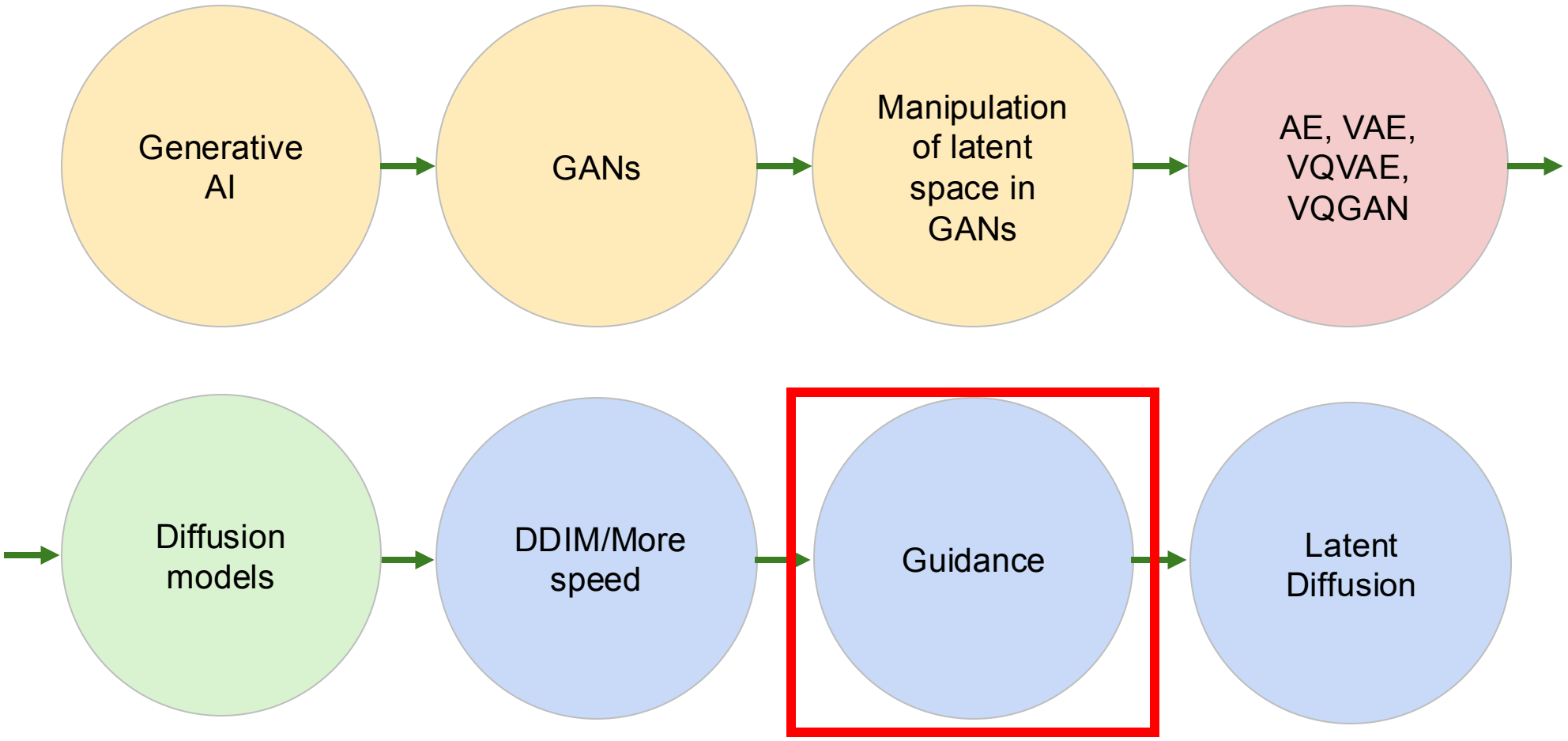


Generalization of DDPMs: Interpolation

x_0^1 $ax_0^1 + (1-a)x_0^2$ x_0^2



Multimodal Generative AI with Diffusion

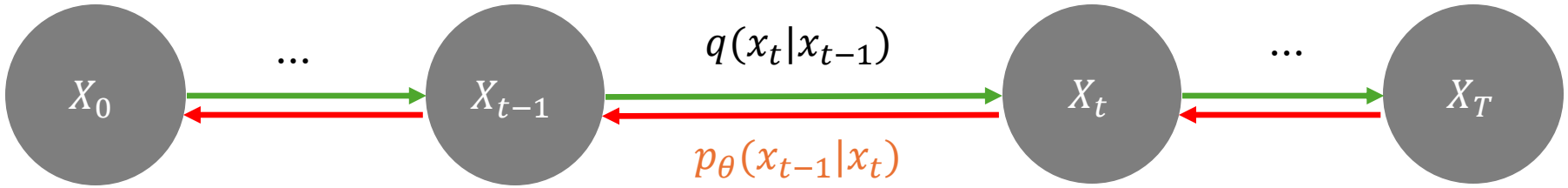


Part VII: Outline

Guidance

- Guided diffusion
- Control the diffusion
- Explicit condition
- Guided diffusion
- Why not guided diffusion?
- Classifier-free guidance
- Negative prompting

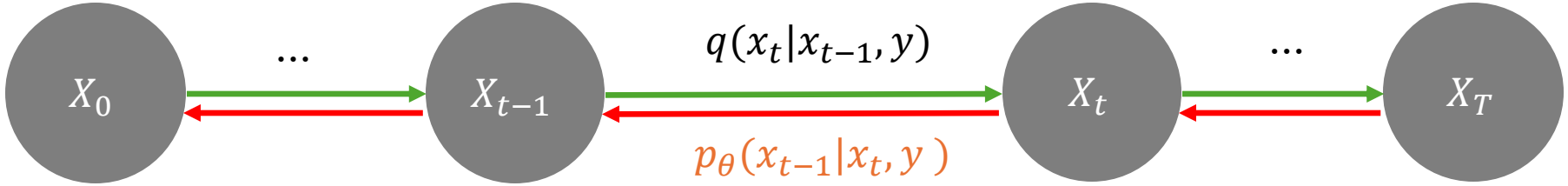
Control the Diffusion Model

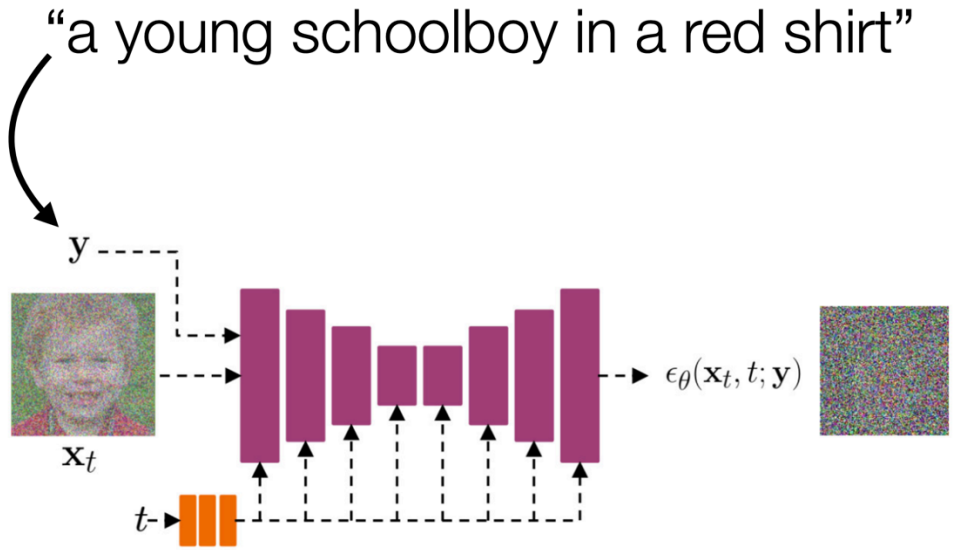


Where is the control?
How did we do with VAE? This sounds a familiar question.

Explicit condition

Control the Diffusion Model: Explicit Condition

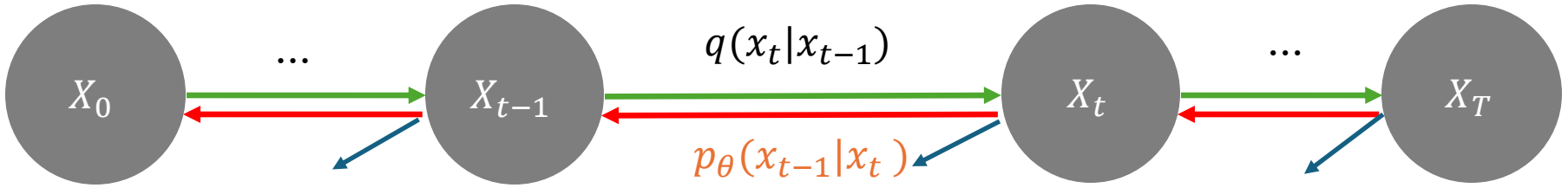




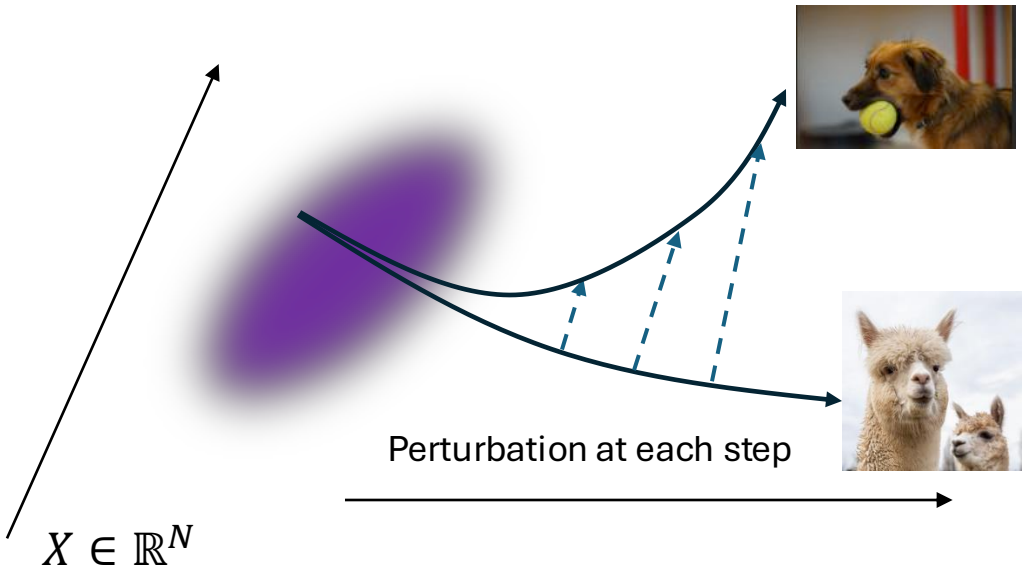
We can add it directly, but is this an effective way? Why?

Guided diffusion

Control the Diffusion Model: Guided Diffusion

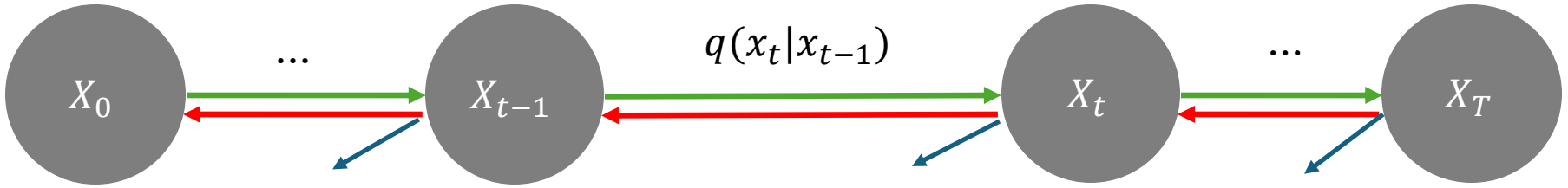


Shift to a dog label!



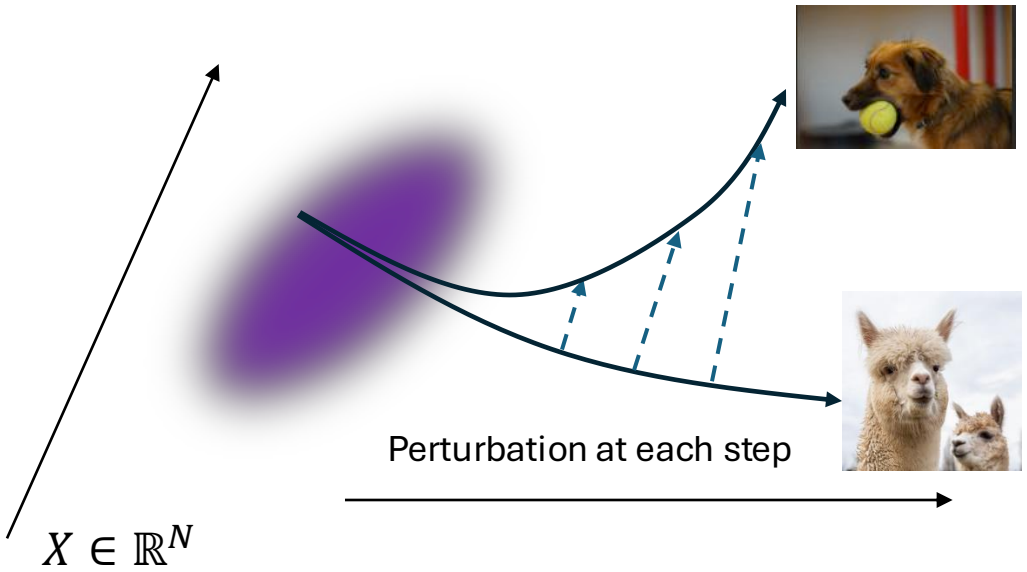
How to do it?

Control the Diffusion Model: Guided Diffusion



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!



It's like a classifier: $p_\phi(y|x)$

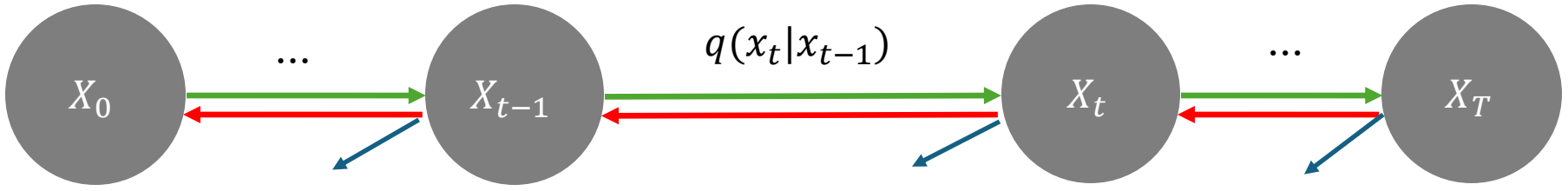
We just look at its gradient:

$$\nabla_x \log p_\phi(y|x)$$

Such that the generated x is similar to the condition label.

In sampling: $\epsilon_\theta(x_t, t) + \nabla_x \log p(y|x)$

Control the Diffusion Model: Guided Diffusion

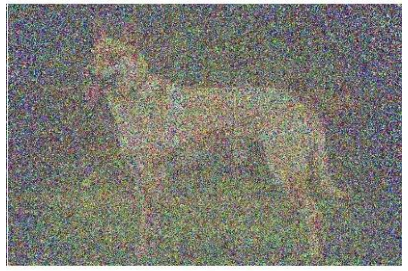


$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

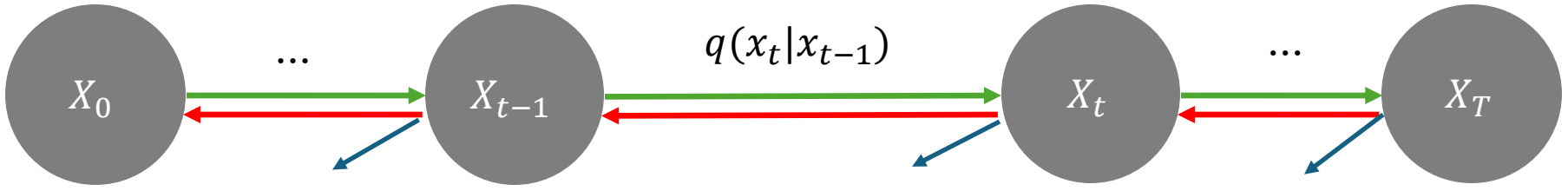
In sampling: $\epsilon_\theta(x_t, t) + \nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

We need to train a classifier: $p_\phi(y|x)$, with the awareness of noise



$p_\phi(y|x)$ A Dog

Control the Diffusion Model: Guided Diffusion



$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

In sampling: $\epsilon_\theta(x_t, t) + \gamma\nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

Label: Corgi

γ = 1

γ = 3

Dhariwal and Nichol, 2021

Guided Diffusion: Nearest Neighbors for Samples

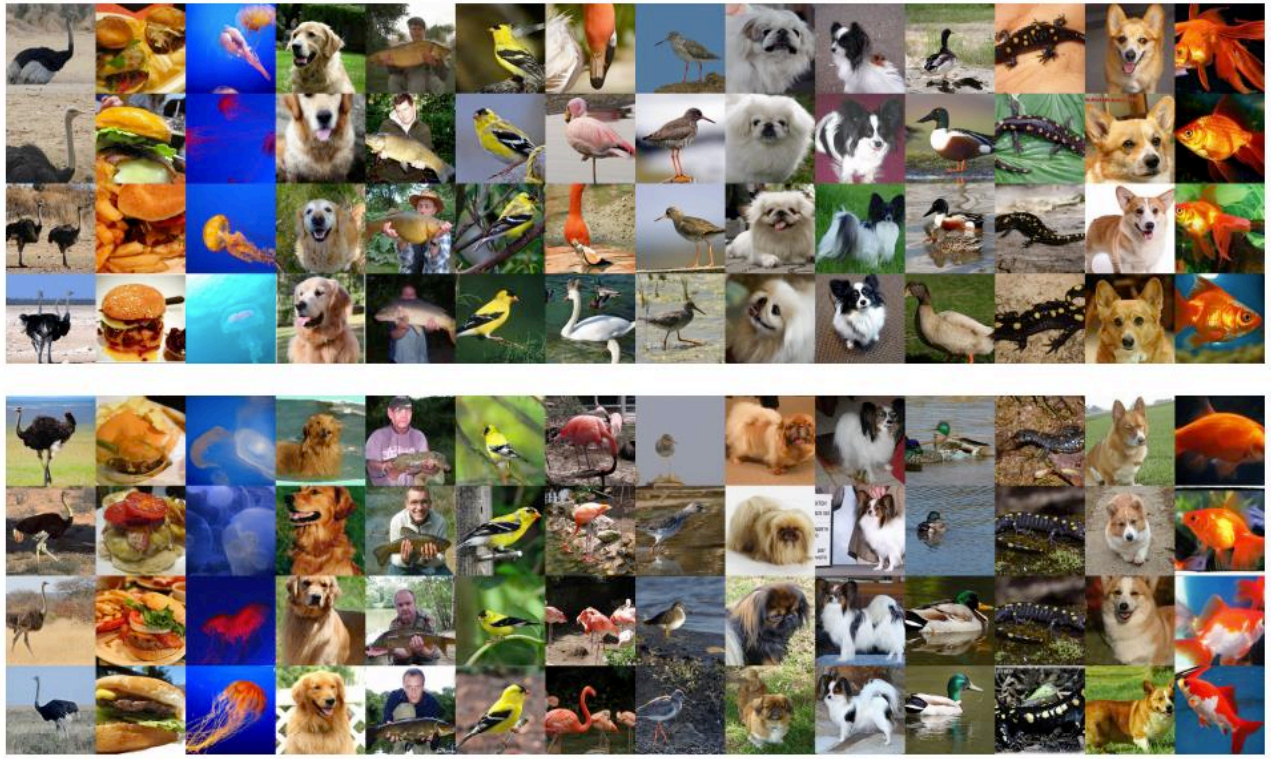
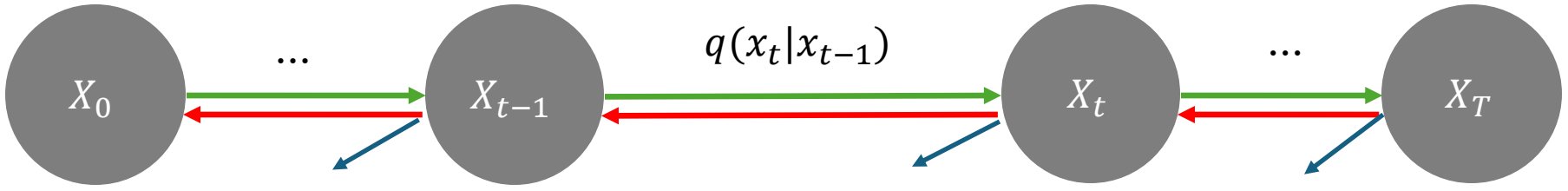


Figure 7: Nearest neighbors for samples from a classifier guided model on ImageNet 256×256. For each image, the top row is a sample, and the remaining rows are the top 3 nearest neighbors from the dataset. The top samples were generated with classifier scale 1 and 250 diffusion sampling steps (FID 4.59). The bottom samples were generated with classifier scale 2.5 and 25 DDIM steps (FID 5.44).

Why not guided diffusion?

Control the Diffusion Model: Guided Diffusion



$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

Shift to a dog label!

In sampling: $\epsilon_\theta(x_t, t) + \gamma\nabla_x \log p_\phi(y|x)$. ← **Guided Diffusion**

What do we NOT like in guided diffusion?

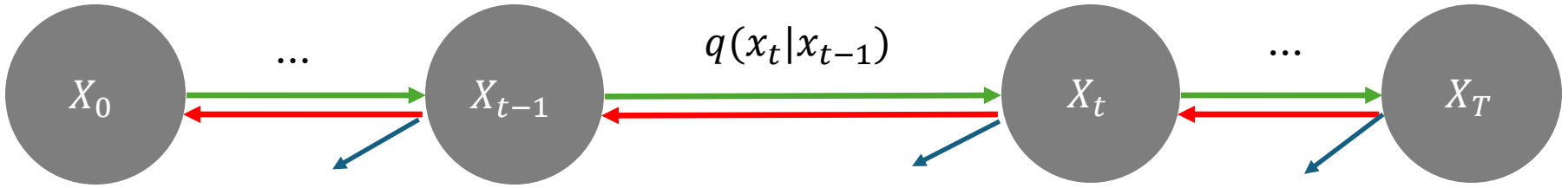
- Need to fine-tune and train a classifier
- Condition can only be label-based, hard to support other conditions like “text input”

Because for text, the classifier $p_\phi(y|x)$ ***does not*** exist.

Dhariwal and Nichol, 2021

Classifier-free guidance

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

At training: $p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$

In sampling: $\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma \nabla_x \log p(y|x)$

$$\nabla_x \log p(y|x) \propto \epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)$$

$$p(y|x) \propto \frac{p(x|y)}{p(x)}$$

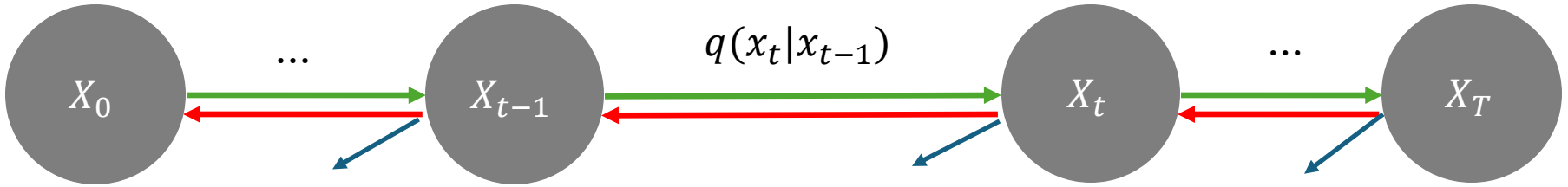
$$\nabla_x \log p(y|x) \propto \nabla_x \log p(x|y) - \nabla_x \log p(x)$$

Thanks to Bayes

Finally: $\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\underbrace{\epsilon_\theta(x_t, t, y)}_{\text{conditional generation}} - \underbrace{\epsilon_\theta(x_t, t)}_{\text{unconditional generation}})$

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\underbrace{\epsilon_\theta(x_t, t, y)}_{\text{conditional generation}} - \underbrace{\epsilon_\theta(x_t, t)}_{\text{unconditional generation}})$$

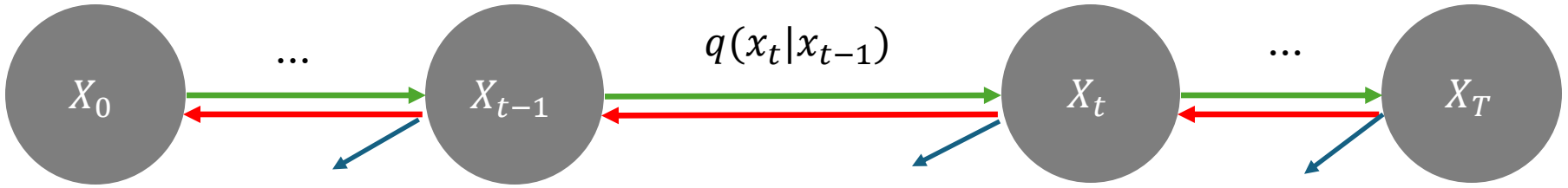
Implicit classifier

How to compute: $\epsilon_\theta(x_t, t, y)$ →
- explicit condition

How to compute: $\epsilon_\theta(x_t, t)$ →
- We randomly set the condition to null (drop-out condition)
- $\epsilon_\theta(x_t, t, y)$ → $\epsilon_\theta(x_t, t, \emptyset)$

Ho and Salimans, 2022

Control the Diffusion Model: Classifier-Free Guidance



$$p_\theta(x_{t-1}|x_t, y) = p_\theta(x_{t-1}|x_t)p_\phi(y|x)$$

Shift to a dog label!

$$\hat{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + \gamma(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t))$$

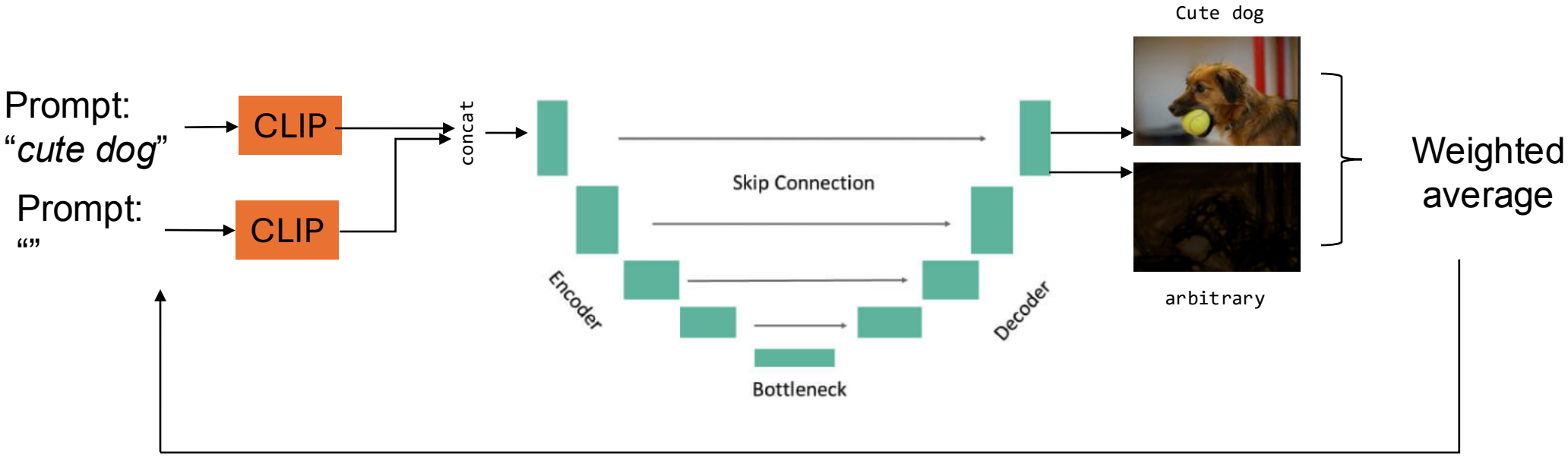
Label: Husky

γ=1

γ=3

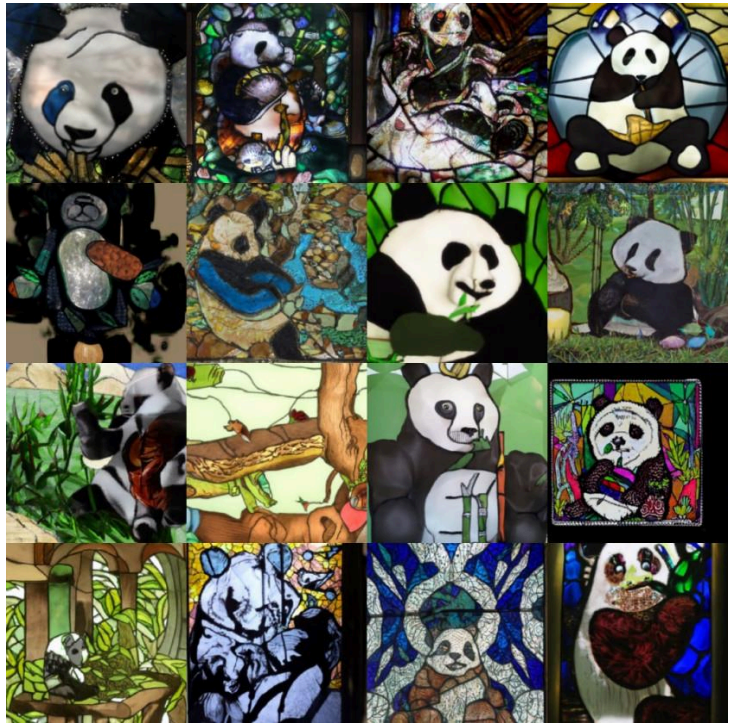
Ho and Salimans, 2022

Classifier free guidance



Classifier free guidance

Control the Diffusion Model: Classifier-Free Guidance



$\gamma = 1$

$\gamma = 3$

Caption: "*A stained glass window of a panda eating bamboo.*"

Control the Diffusion Model: Classifier-Free Guidance

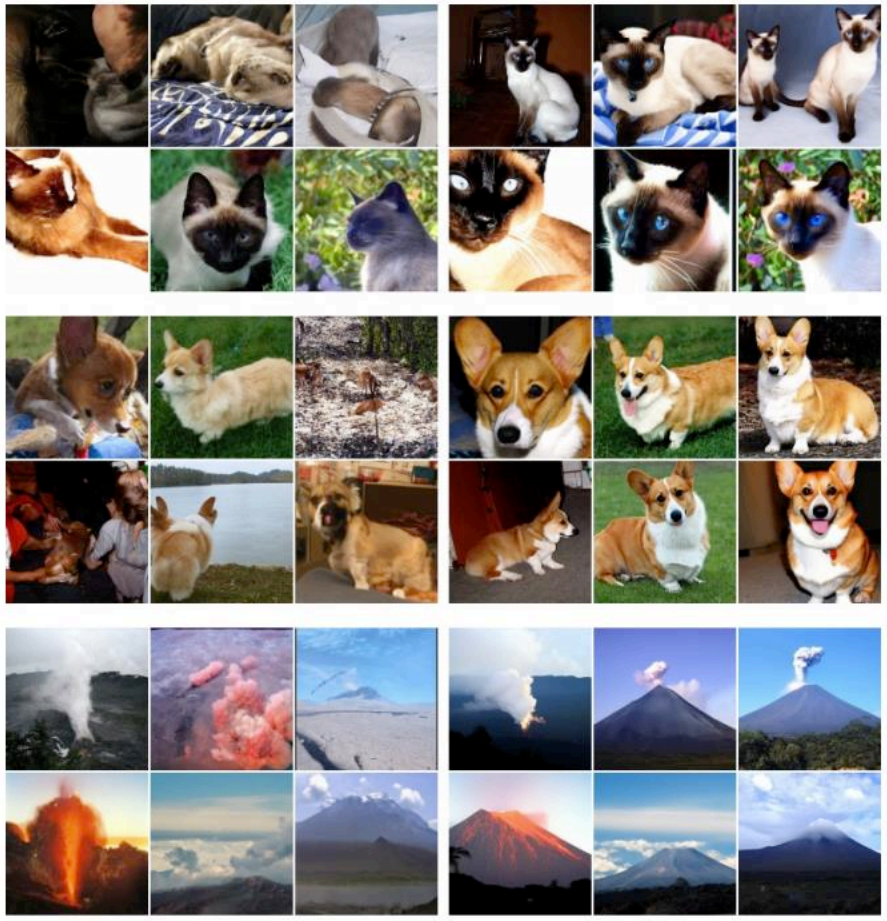


Figure 3: Classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$. Interestingly, strongly guided samples such as these display saturated colors. See Fig. 8 for more.

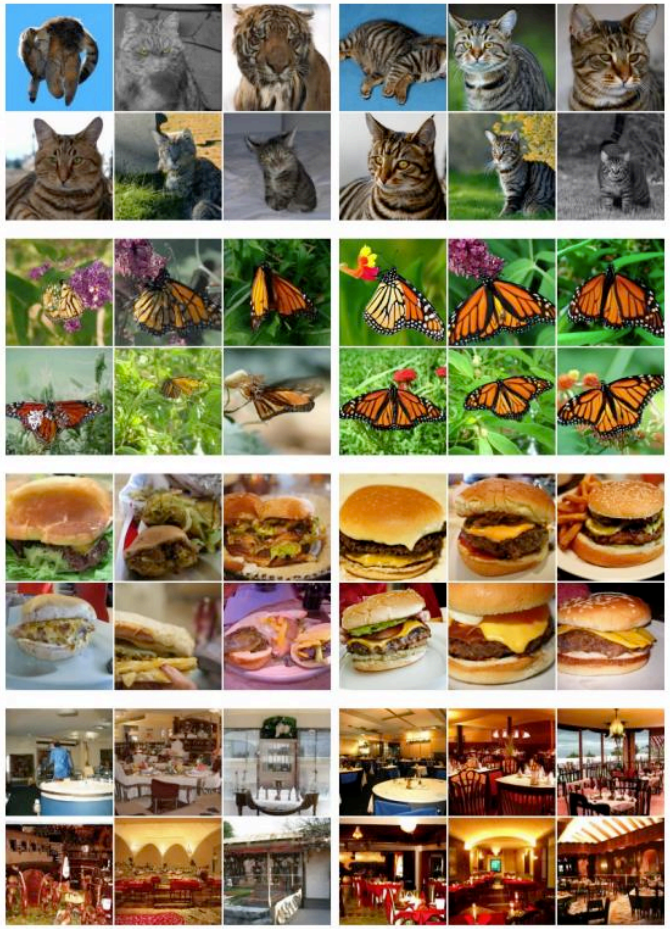
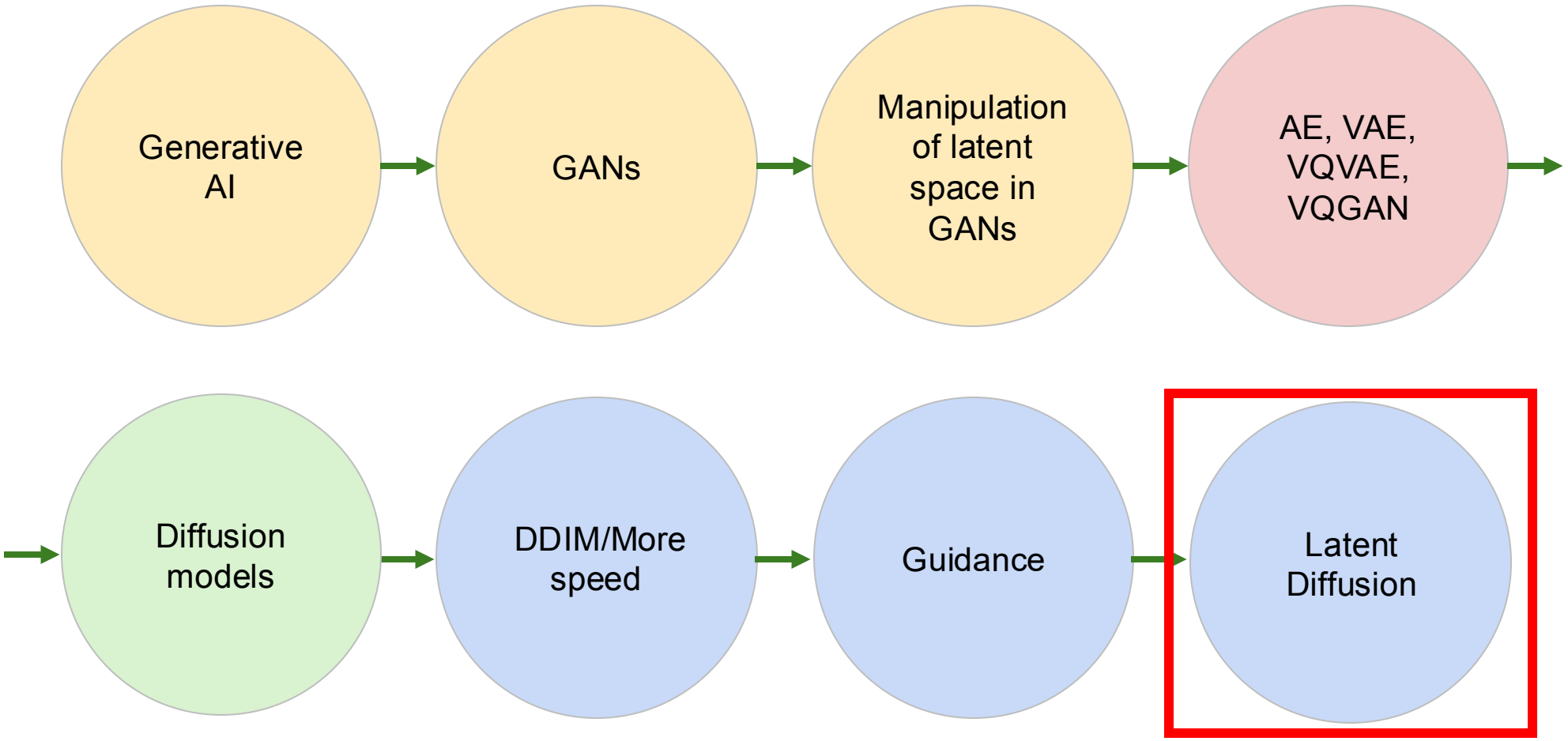
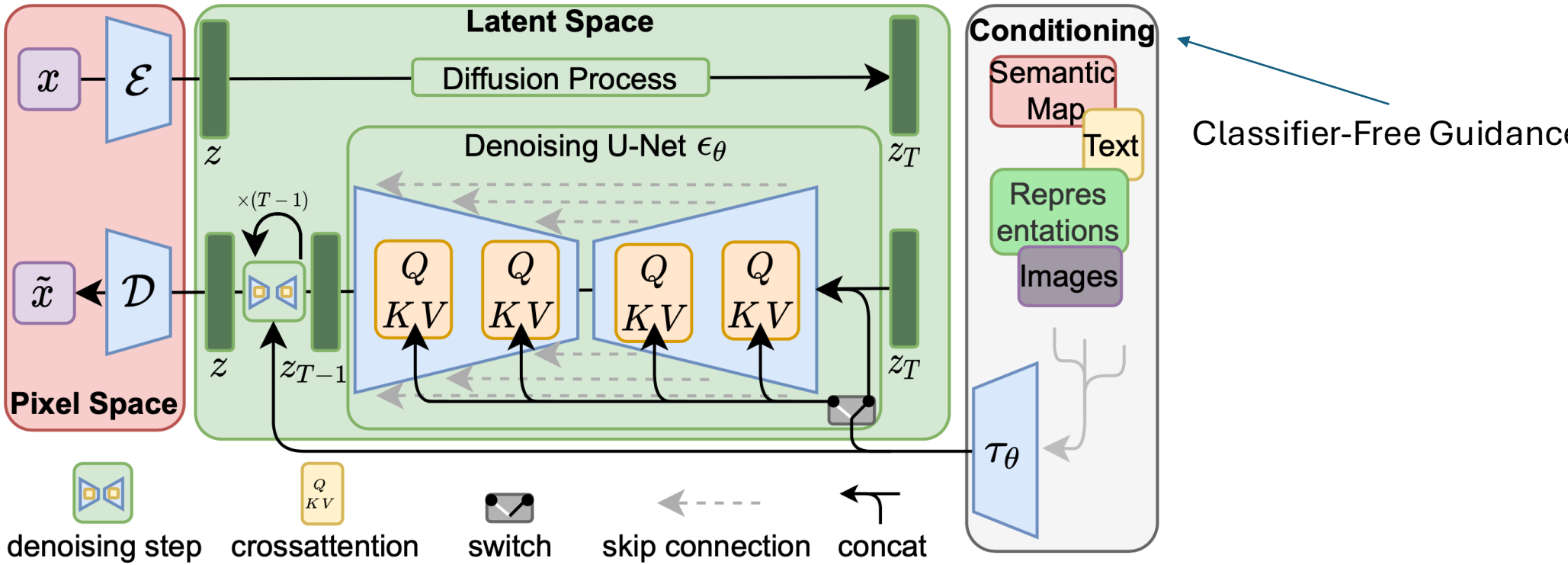


Figure 8: More examples of classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$.

Multimodal Generative AI with Diffusion



Latent Diffusion Model



Instead of denoising on pixels, we could denoise on latent space with could be constructed from a VAE/VQGAN.

Latent Diffusion Model

Thank you