CS 4824/ECE 4424: GAN and Diffusion

Acknowledgement:
Many of these slides are derived from Tom Mitchell, Pascal Poupart, Pieter Abbeel, Eric Eaton, Carlos Guestrin, William Cohen, and Andrew Moore.

Generative networks

- Neural networks are typically used for classification or regression
  - Input: data
  - Output: class or prediction
- Can we design neural networks that can generate data?
  - Input: random vector
  - Output: data

Generative networks

- Several types of generative networks

  - Boltzmann machines
  - Sigmoid belief networks
  - Variational autoencoders
  - **Generative adversarial networks**
  - Generative moment matching networks
  - Sum-product networks
  - Normalizing flows
  - **Diffusion**
  - Flow-matching
  - …

Generative Adversarial Networks

- Approach based on game theory

- Two networks:
  - Generator $g(\boldsymbol{z}; \boldsymbol{W}_g) \rightarrow \boldsymbol{x}$
  - Discriminator $d(\boldsymbol{x}; \boldsymbol{W}_d) \rightarrow \Pr(\boldsymbol{x}\ is\ real)$ x

- Objective

$$\min_{W_g} \max_{W_d} \Pr(X_n \text{ is real}; W_d) + \Pr(g(z_n; W_g) \text{ is fake}; W_d)$$

$$= \min_{W_d} \max_{W_d} d(x_n; W_d) + (1 - d(g(z_n; W_g); W_d))$$

Generative Adversarial Networks

- Approach based on game theory

- Two networks:
  - Generator $g(\boldsymbol{z}; \boldsymbol{W}_g) \rightarrow \boldsymbol{x}$
  - Discriminator $d(\boldsymbol{x}; \boldsymbol{W}_d) \rightarrow \Pr(\boldsymbol{x}\ is\ real)$

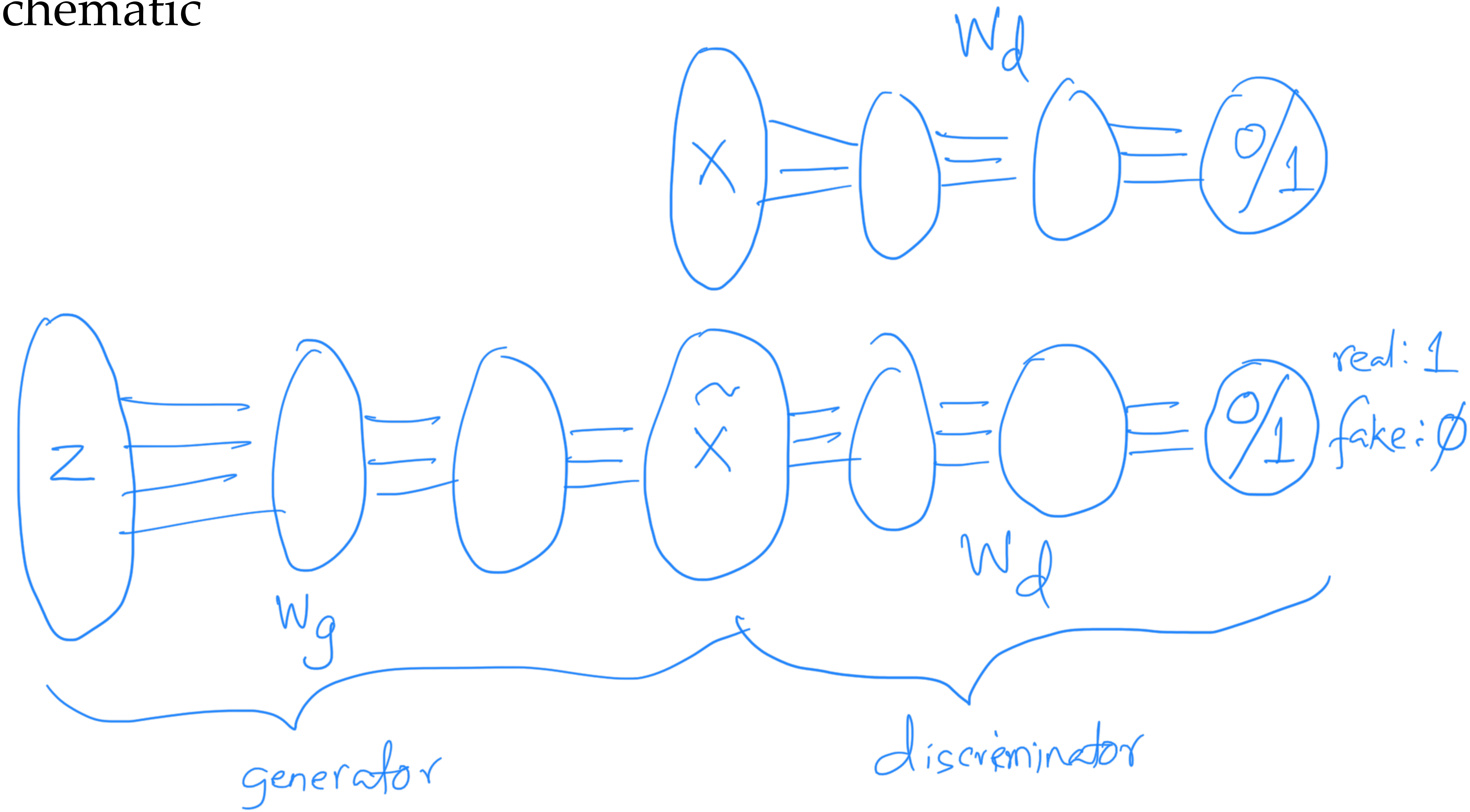
- Objective

$$\min_{\boldsymbol{W}_g} \boxed{\max_{\boldsymbol{W}_d} \sum_n \log \Pr(\boldsymbol{x}_n\ is\ real; \boldsymbol{W}_d) + \log \Pr\left(g\left(\boldsymbol{z}_n; \boldsymbol{W}_g\right) is\ fake; \boldsymbol{W}_d\right)}$$

$$\equiv \min_{\boldsymbol{W}_g} \max_{\boldsymbol{W}_d} \sum_n \log d(\boldsymbol{x}_n; \boldsymbol{W}_d) + \log\left(1 - d\left(g(\boldsymbol{z}_n; \boldsymbol{W}_g); \boldsymbol{W}_d\right)\right)$$

Generative Adversarial Networks

- Schematic



GAN training

- We have a min-max optimization
  - Optimize the discriminator by stochastic gradient ascent
    - Optimize the generator by stochastic gradient descent

GAN training

- Repeat until convergence
  - For k steps do
    - Sample $\mathbf{z}_1, \ldots, \mathbf{z}_N$ from $\Pr(\mathbf{z})$
    - Sample $\mathbf{x}_1, \ldots, \mathbf{x}_N$ from training set
    - Update discriminator by ascending its stochastic gradient

$$\boxed{\nabla_{\boldsymbol{W}_d}\left(\frac{1}{N}\sum_{n=1}^{N}\left[\log d(\boldsymbol{x}_n;\boldsymbol{W}_d)+\log\left(1-d\left(g\left(\boldsymbol{z}_n;\boldsymbol{W}_g\right);\boldsymbol{W}_d\right)\right)\right]\right)}$$

- Sample $\mathbf{z}_1, \ldots, \mathbf{z}_N$ from $\Pr(\mathbf{z})$
- Update generator by descending its stochastic gradient

$$\boxed{\nabla_{\boldsymbol{W}_g}\left(\frac{1}{N}\sum_{n=1}^{N}\log\left(1-d\left(g\left(\boldsymbol{z}_n;\boldsymbol{W}_g\right);\boldsymbol{W}_d\right)\right)\right)}$$

GAN training

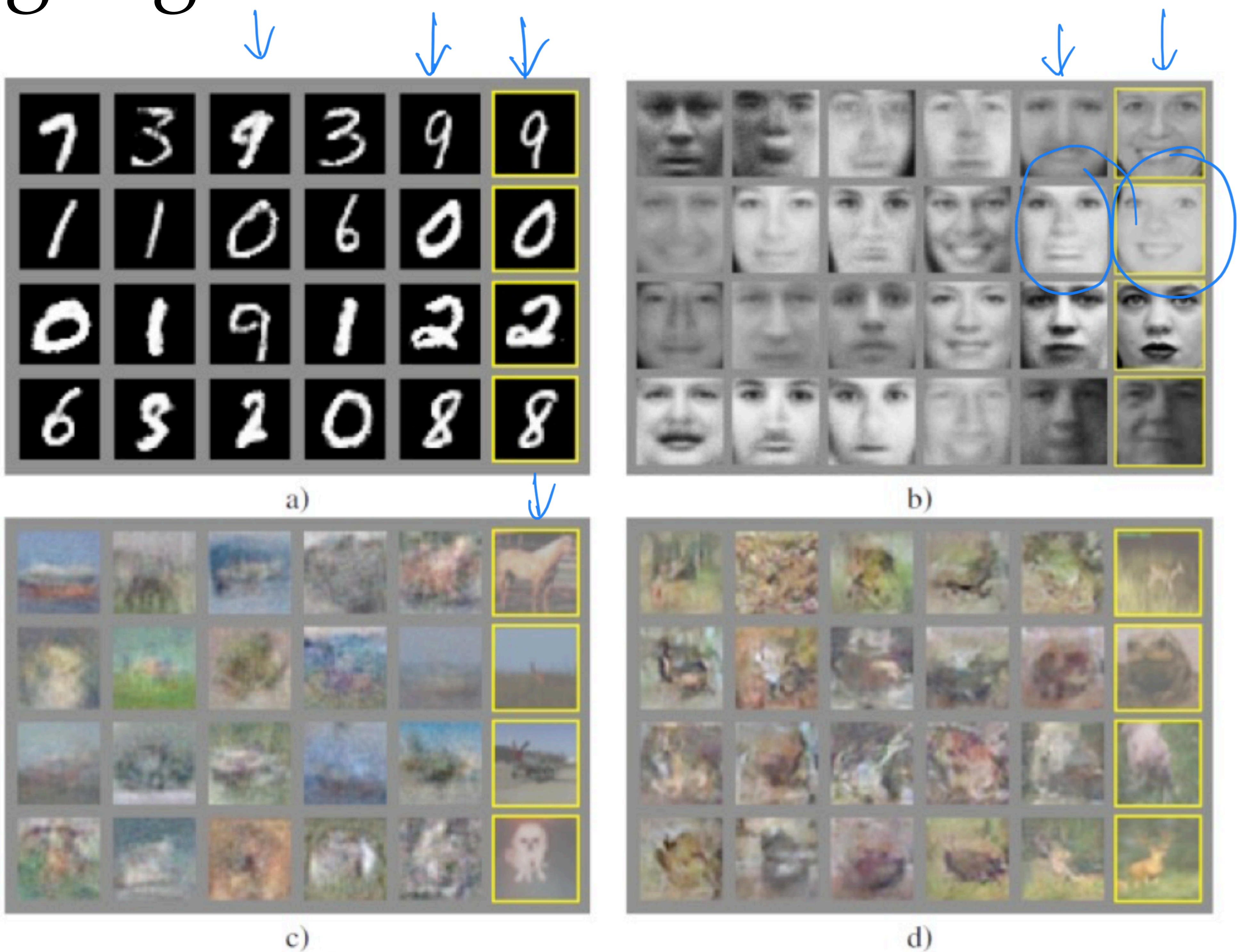
- In the limit (with sufficiently expressive networks, sufficient data and global convergence)

$$\boxed{\Pr(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{W}_g)} \rightarrow \textit{true data distribution}$$

$$\Pr(\boldsymbol{x}\ \textit{is real}; \boldsymbol{W}_d) \rightarrow 0.5 \text{ (for real and fake data)}$$

- Problems in practice:
  - Imbalance: one network may dominate the other
  - Local convergence

Images generated with GANs training



- Right columns are nearest neighbour training examples of adjacent column

Diffusion

- Diffusion models perform *incremental* updates to generate data from white noise.
- Why increment? It’s like turning the direction of a giant ship. You need to turn the ship slowly towards your desired direction or otherwise you will lose control.
- The philosophy: “**Bend one inch at a time.**”

Denoising Diffusion Probabilistic Model (DDPM)

- Two phases:
  - Forward Diffusion (Noising)
  - Reverse Diffusion (Denoising)

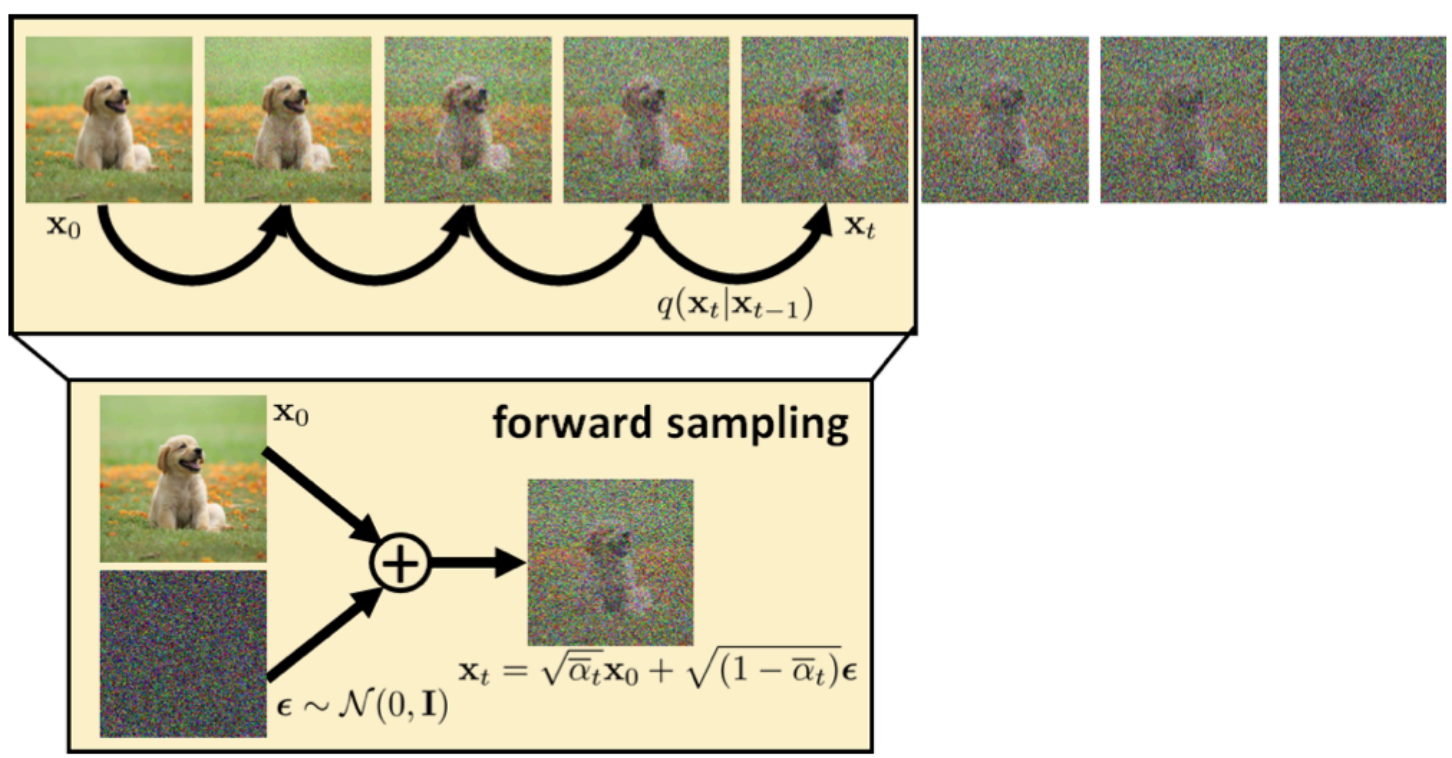
- In between the two phases: train DDPM

Forward Diffusion in DDPM

- The goal of forward diffusion is to generate the intermediate variables by using

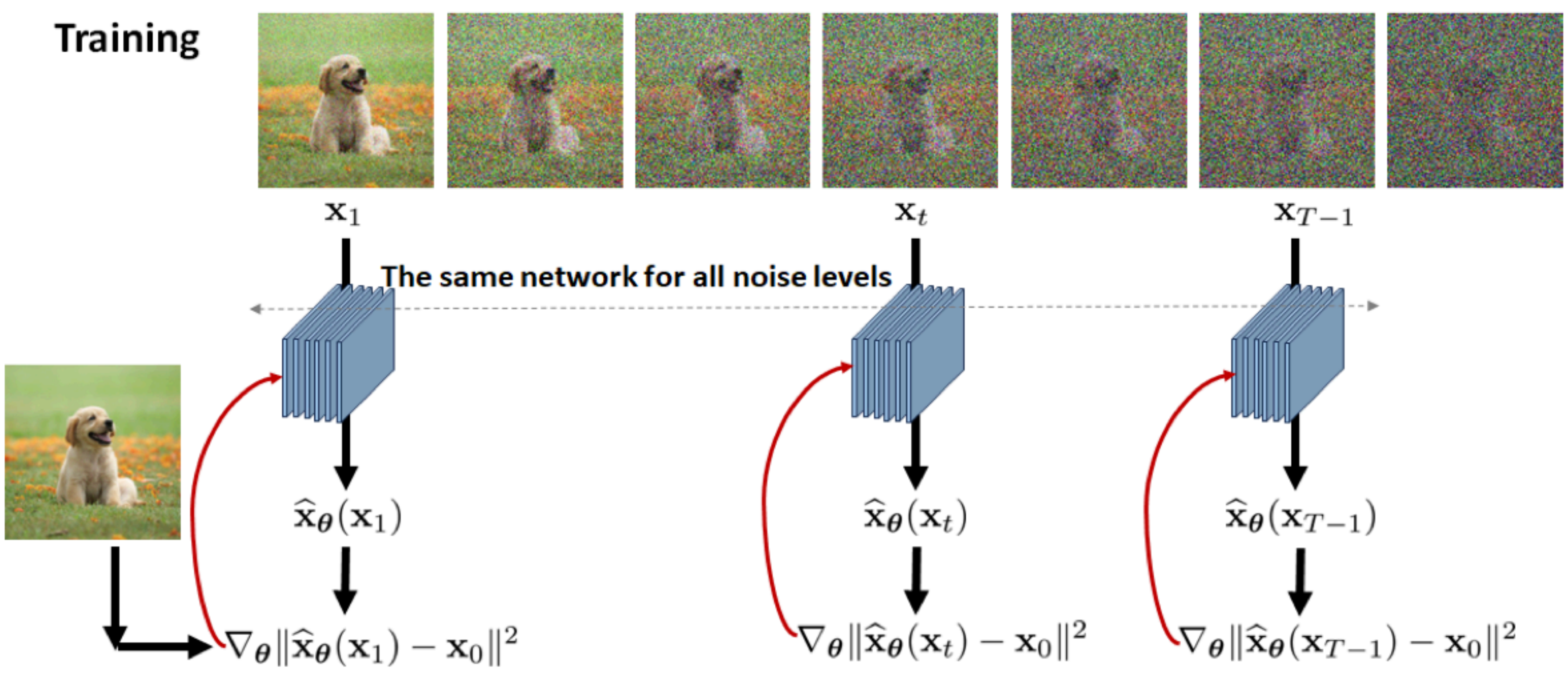
$$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t \,|\, \sqrt{\overline{\alpha}_t}\mathbf{x}_0, \;\; (1-\overline{\alpha}_t)\mathbf{I}), \qquad t = 1, \ldots, T-1.$$

- The forward diffusion does not require any training. Given the clean image, we can run the forward diffusion.



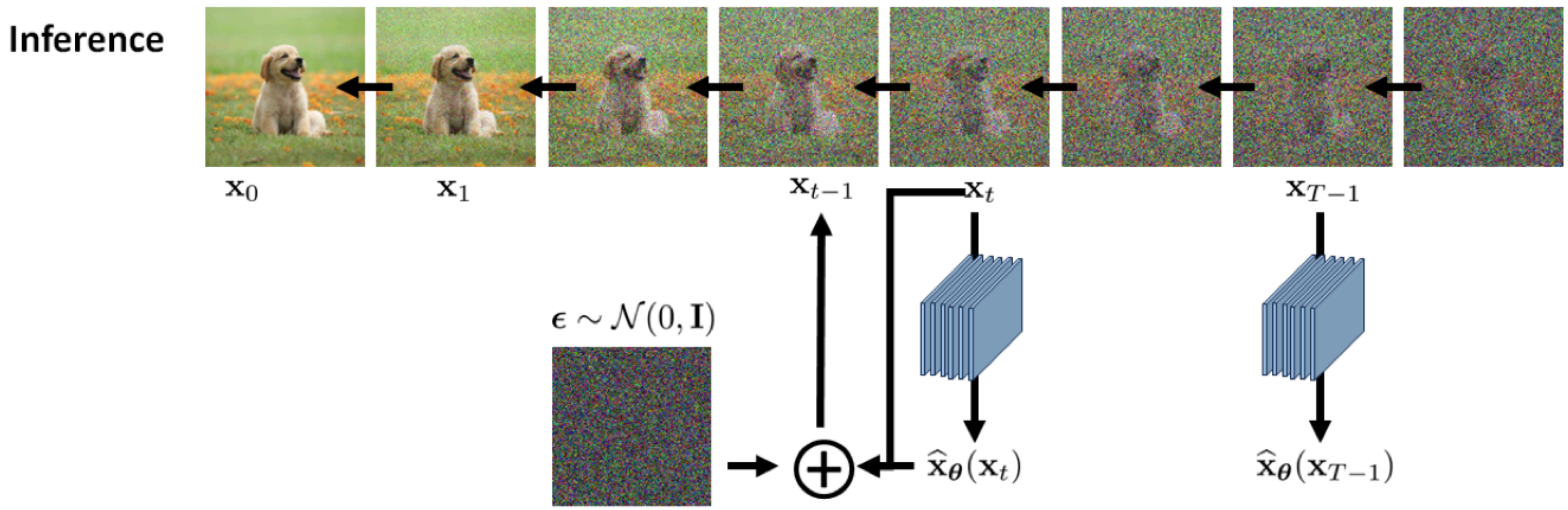
Training DDPM

- The goal is to train a denoiser which takes the clean and noisy images and back-propagates the gradient of the loss is to update the parameters of a neural network.

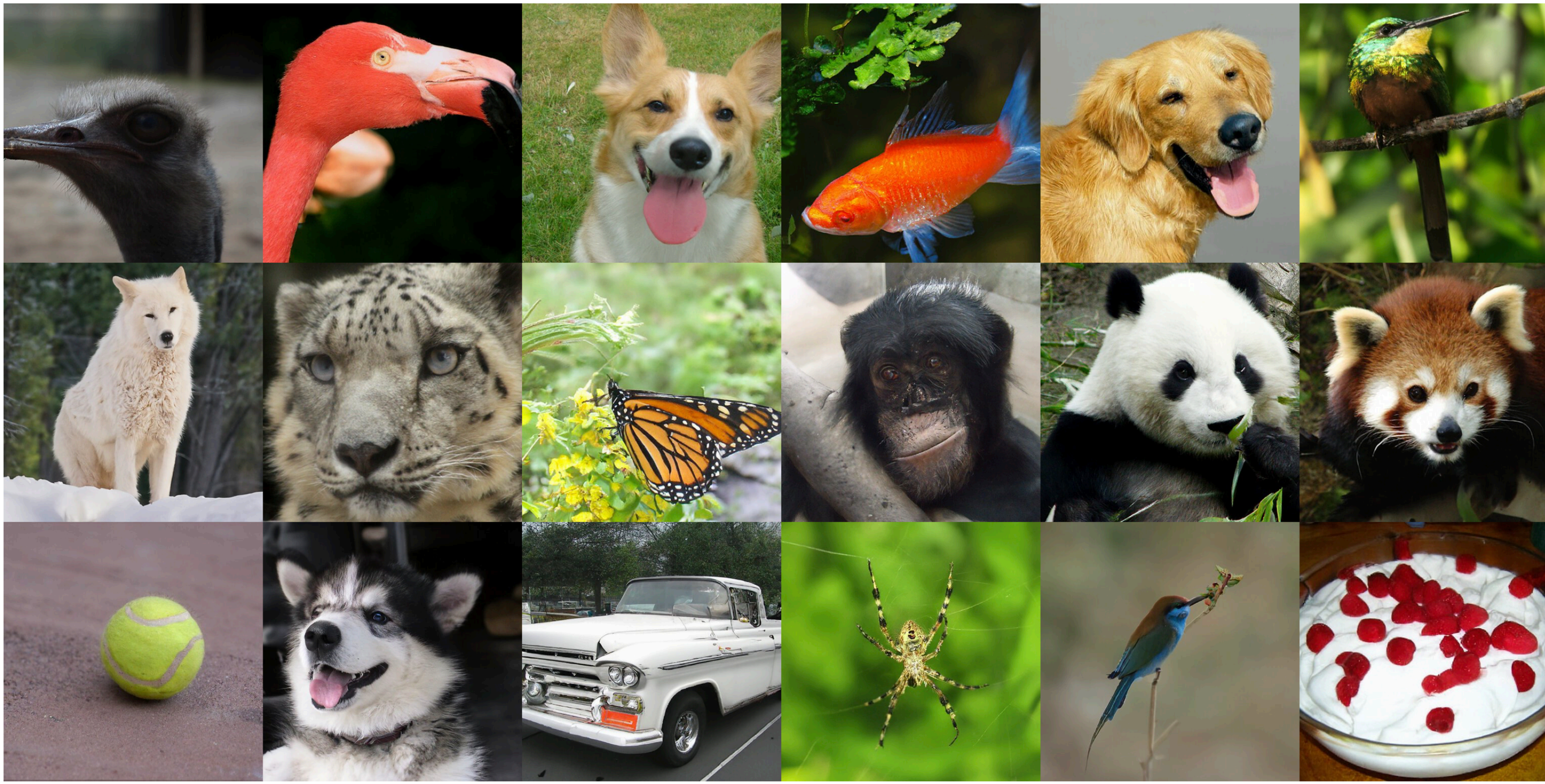


Inference of DDPM – the Reverse Diffusion

- Sequentially run the denoiser T times from white noise back to the generated image.



Diffusion Models Beat GANs on Image Synthesis



https://arxiv.org/pdf/2105.05233

- Selected samples from OpenAI best ImageNet 512×512 model (FID 3.85)