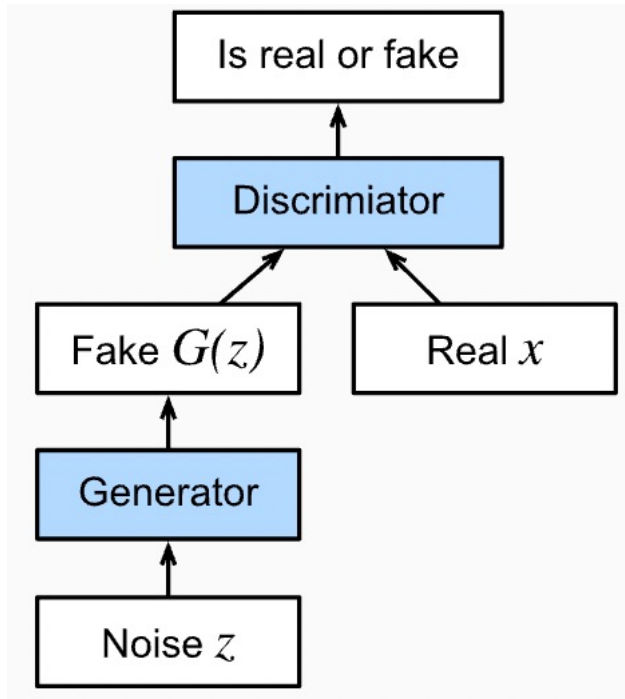


Generative Adversarial Networks

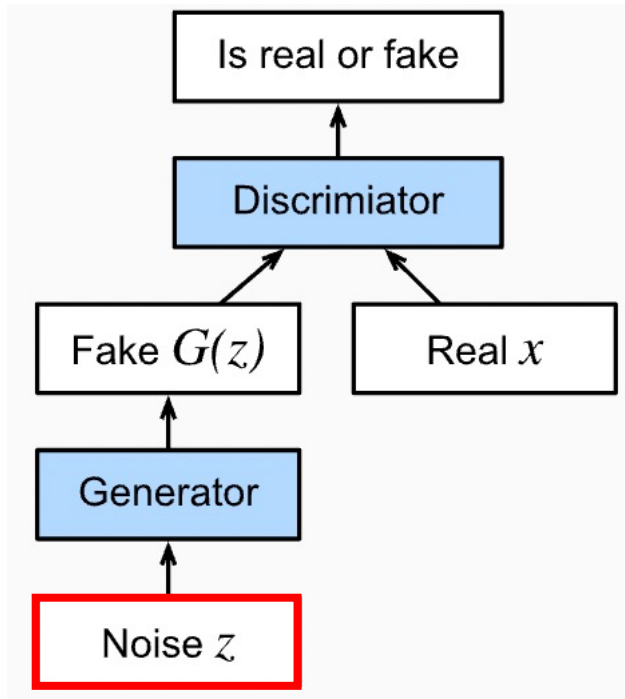
Neural Networks Design And Application

Generative adversarial networks



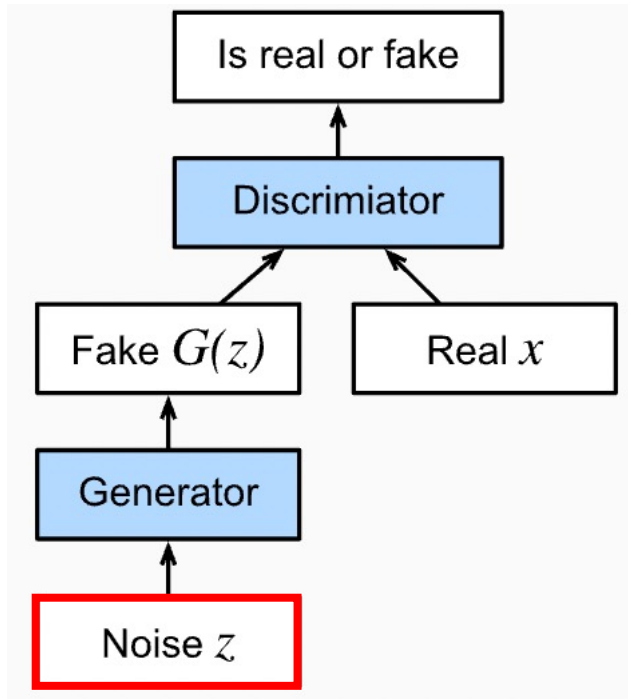
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



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

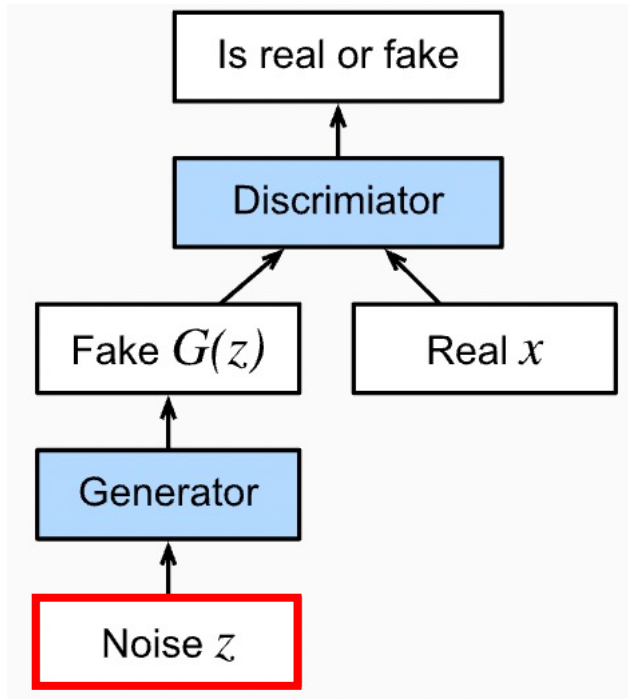
Generative adversarial networks



e.g., sampled from Gaussian $\sim (0,1)$

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

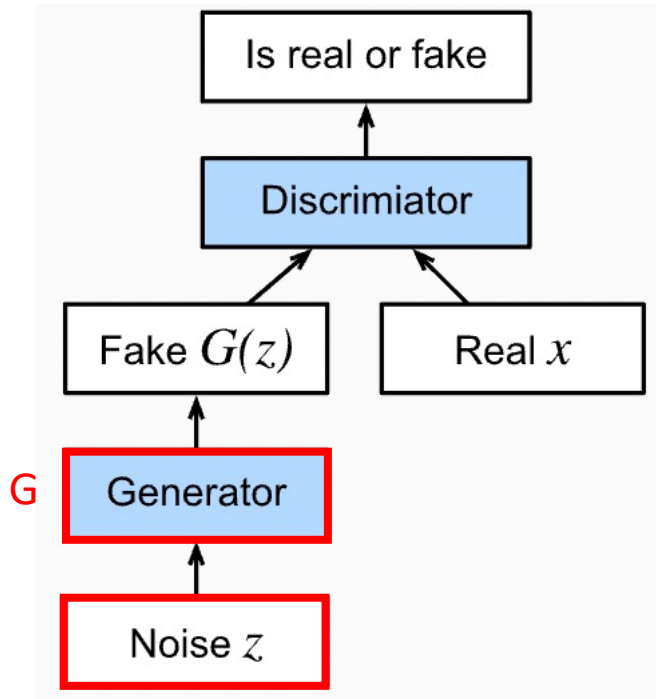
Generative adversarial networks



e.g., sampled from $\text{Gaussian} \sim (0,1)$
(Latent variable)

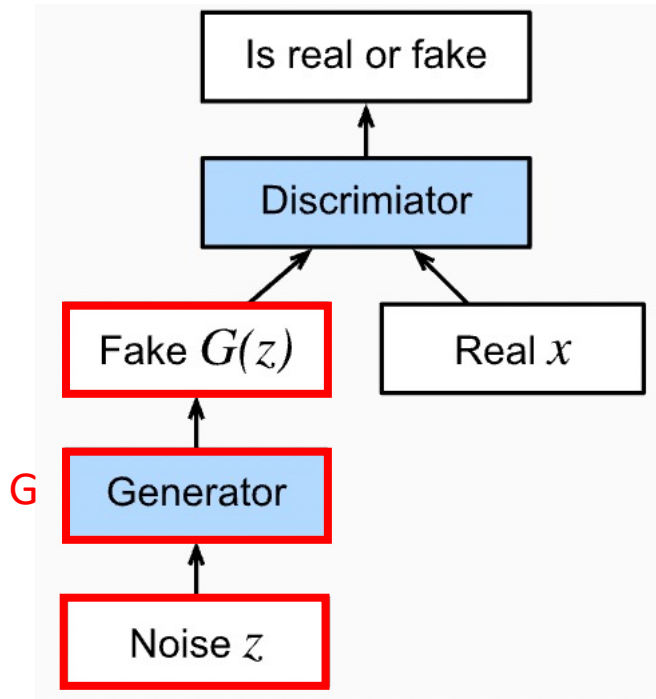
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



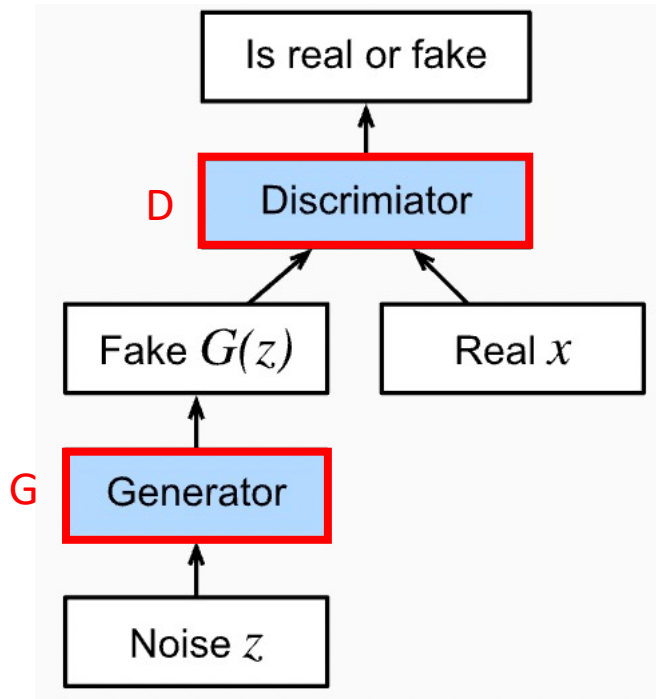
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



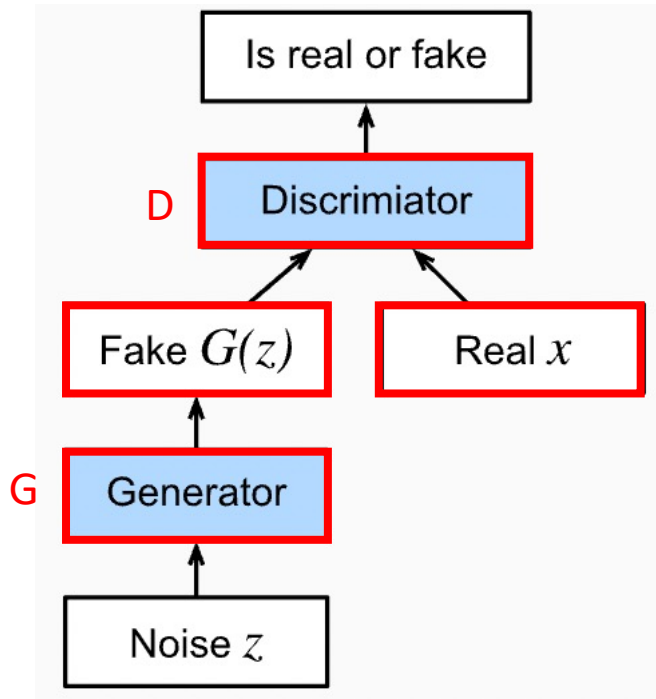
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



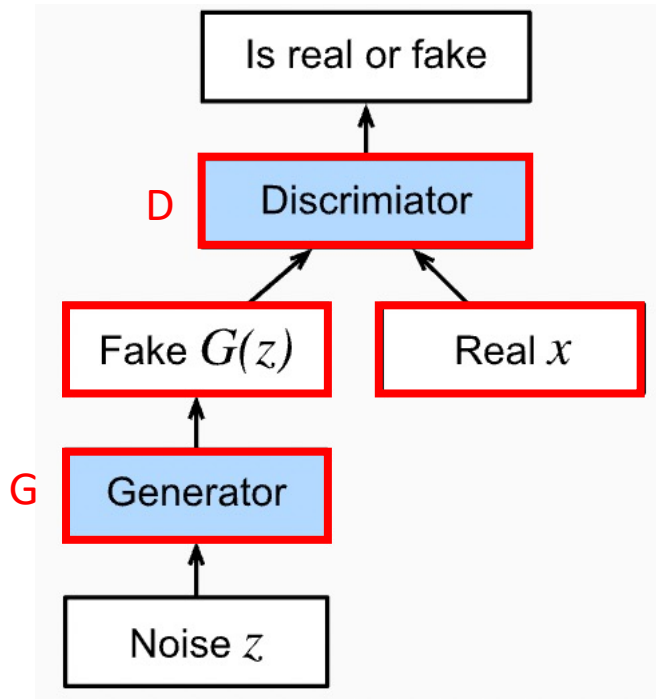
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



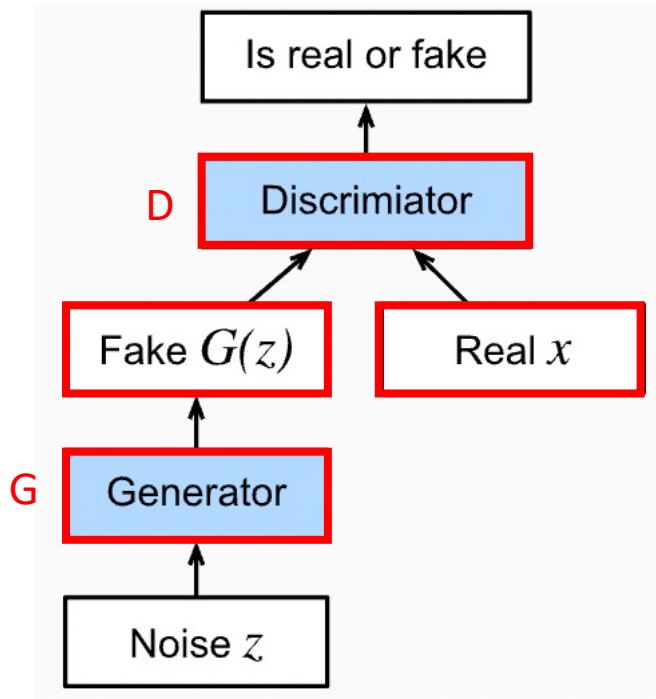
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



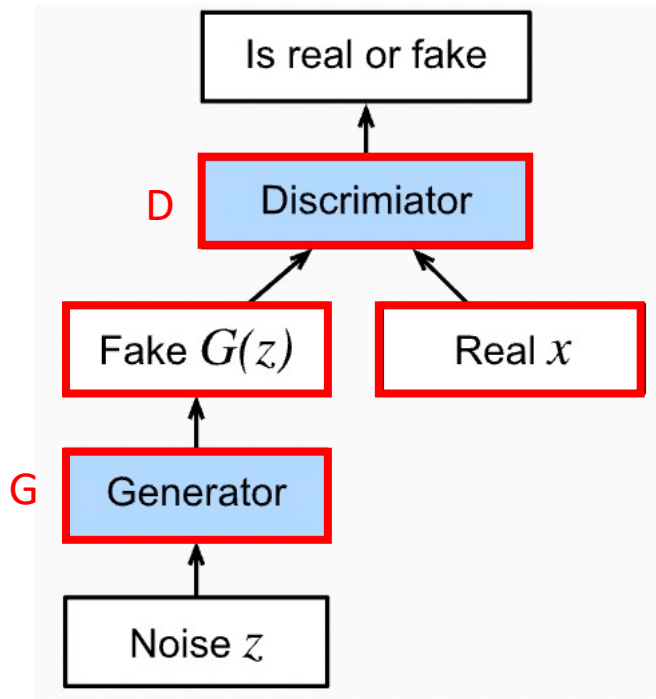
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

Generative adversarial networks



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

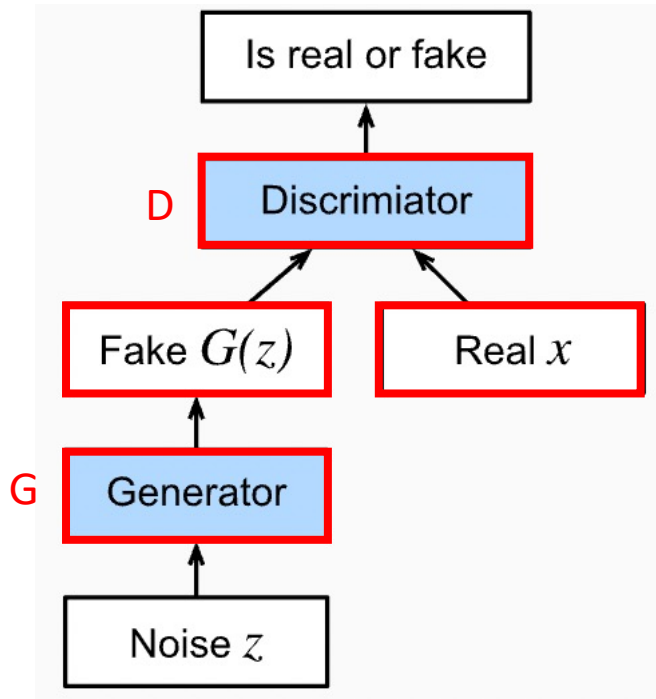
Generative adversarial networks



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

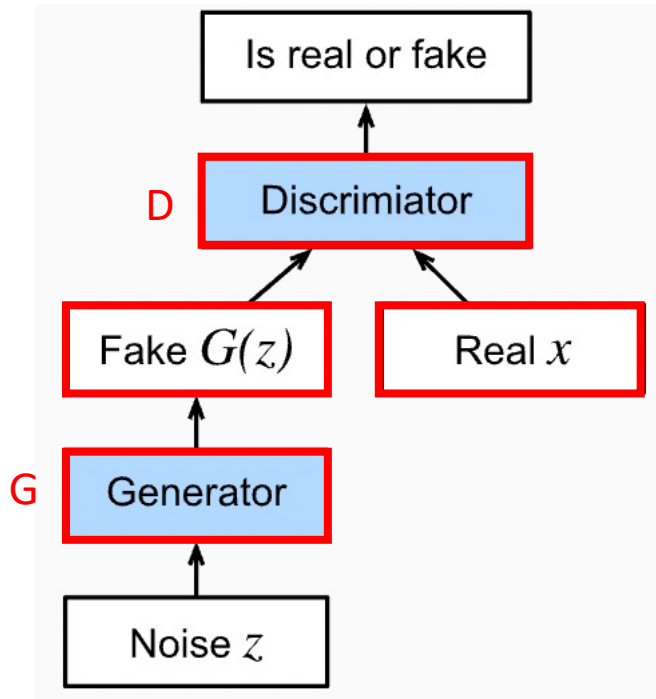
Built on p_{data}

Generative adversarial networks



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

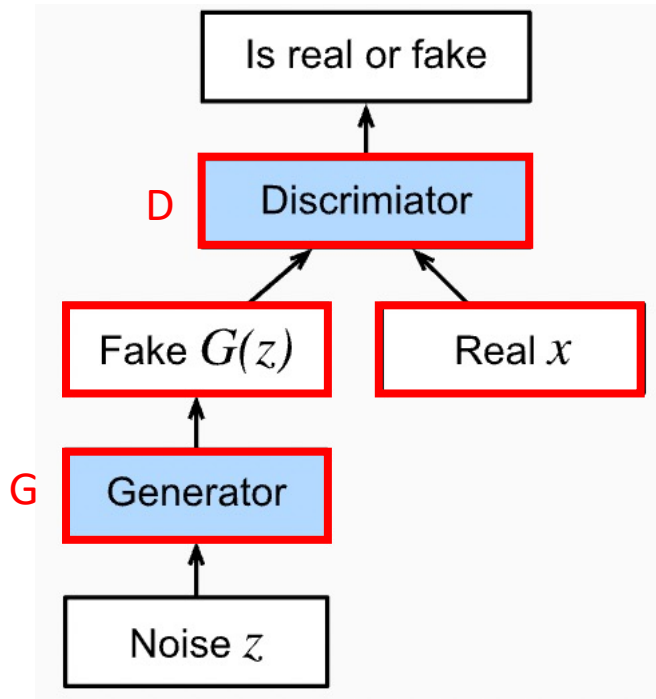
Generative adversarial networks



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

MLE for 1-discriminator

Generative adversarial networks

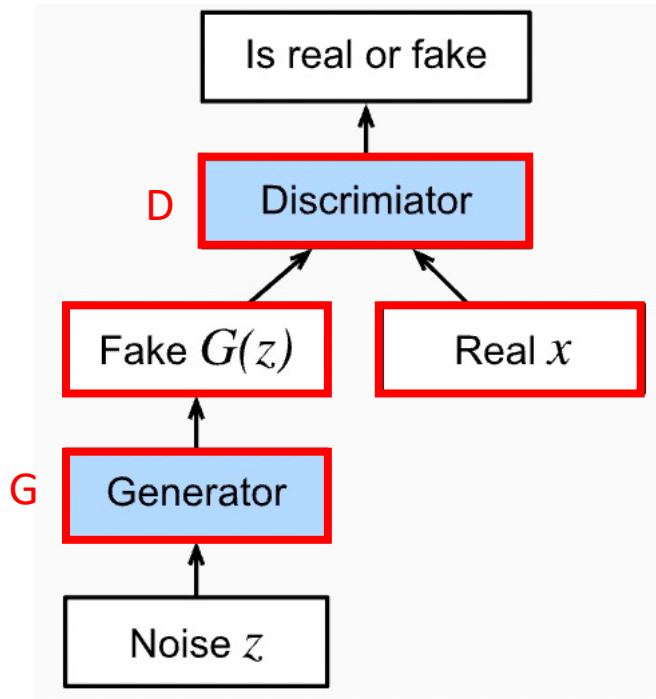


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] .$$

MLE for 1-discriminator

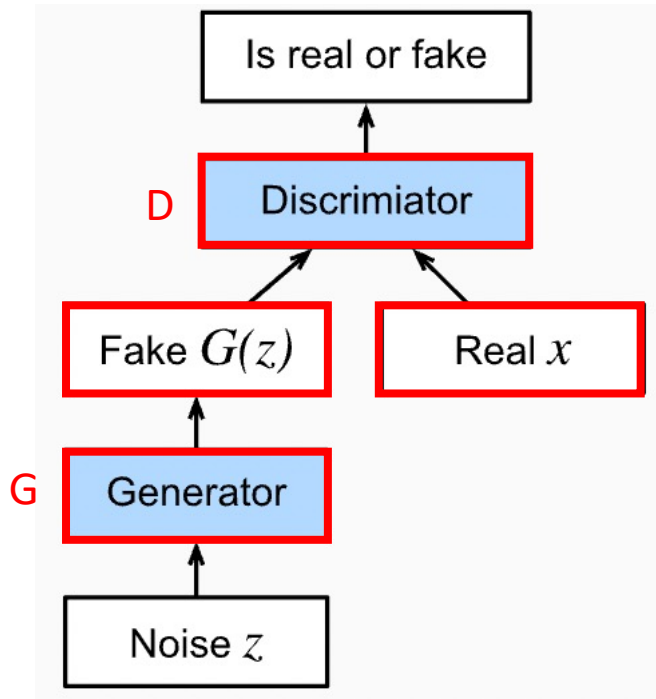
Built on $G(z)$

Generative adversarial networks



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

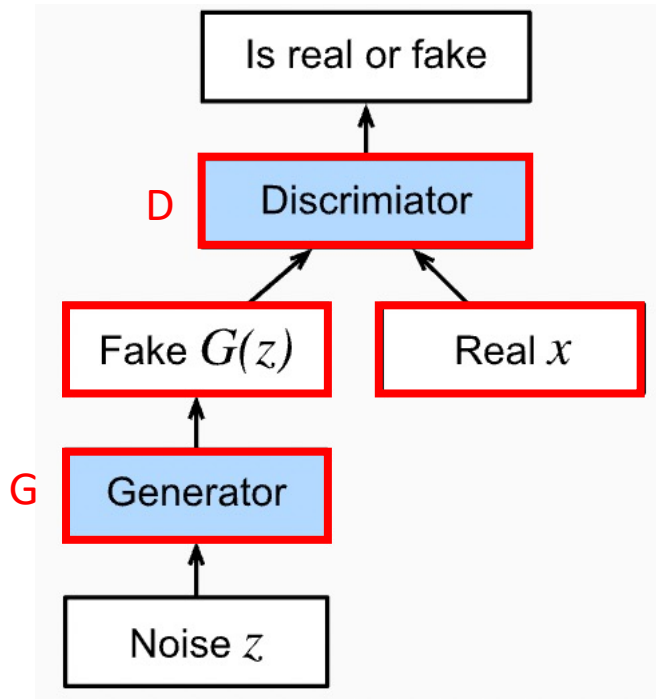
Generative adversarial networks



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

$$\begin{aligned} &w.r.t. G: \\ &\min \log(1 - D(G(z))) \end{aligned}$$

Generative adversarial networks

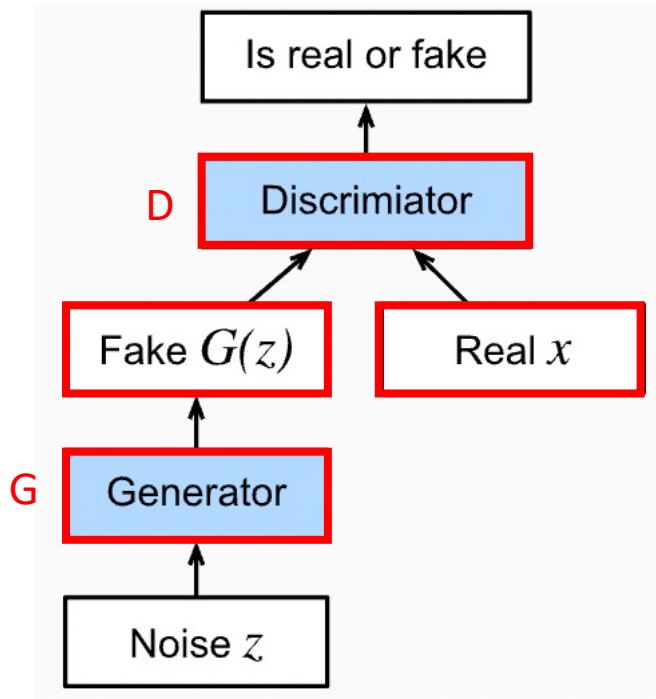


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

w.r.t. G :
 $\min \log(1 - D(G(z)))$

D : output probabilities

Generative adversarial networks

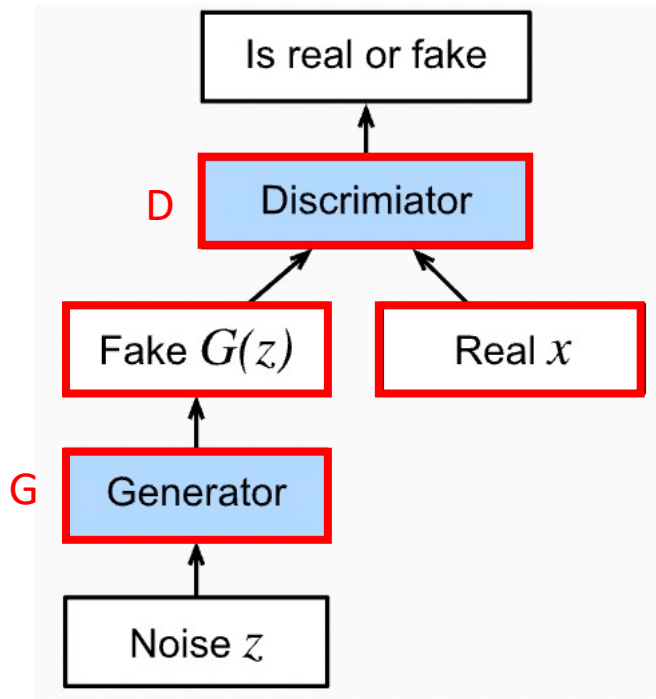


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

w.r.t. G :
 $\min \log(1 - D(G(z)))$

D : output probabilities
 \rightarrow in $[0, 1]$

Generative adversarial networks



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

w.r.t. G :

$$\min \log(1 - D(G(z)))$$

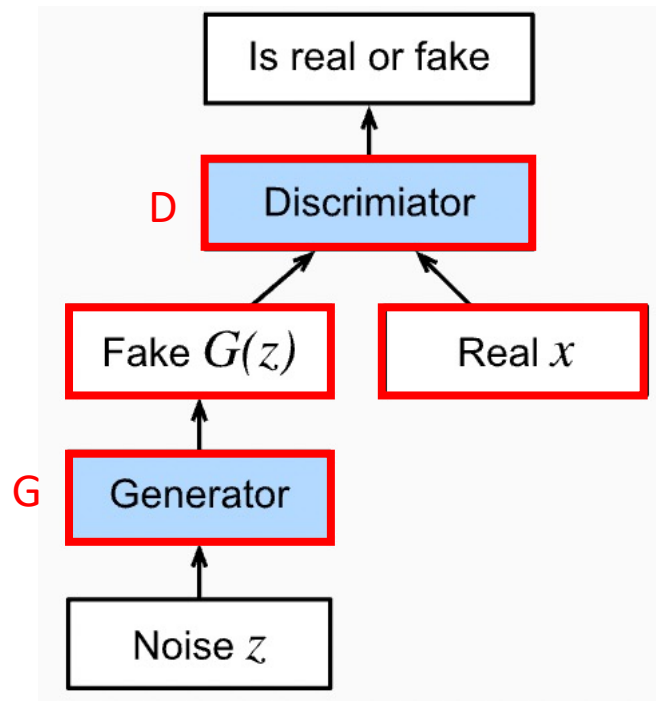
↓

D : output probabilities
→ in $[0, 1]$

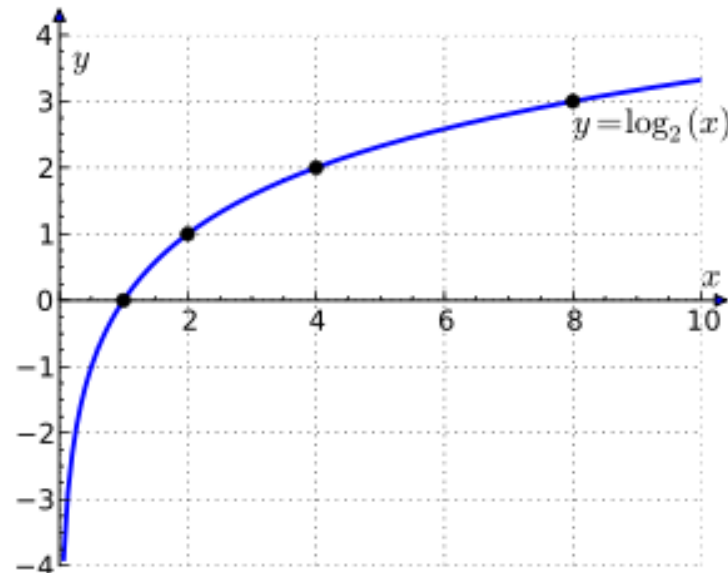
↓

$$D(G(z)) \rightarrow 1$$

Generative adversarial networks

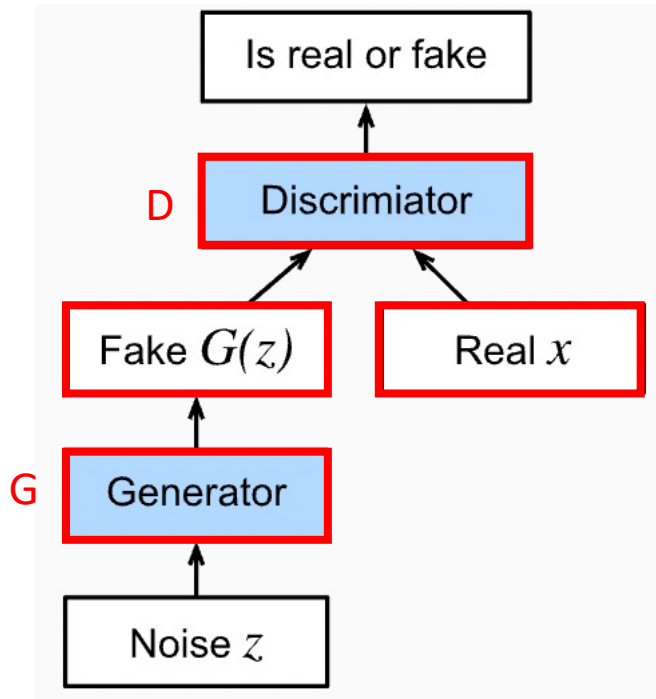


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

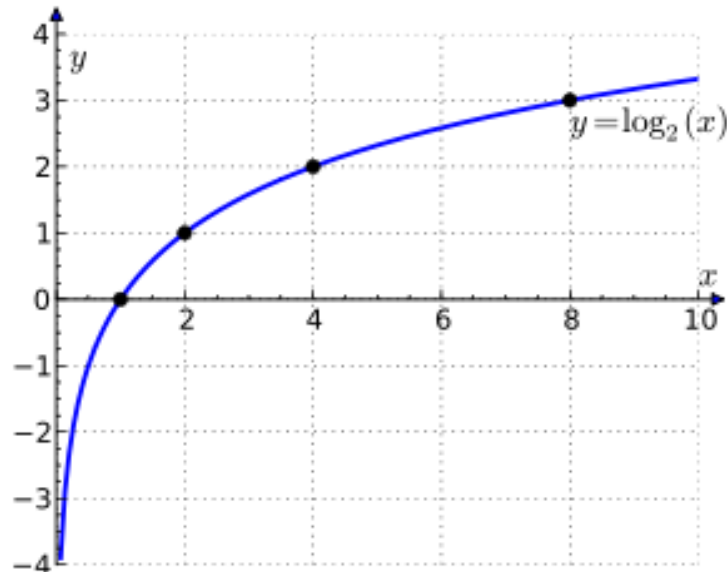


w.r.t. G :
 $\min \log(1 - D(G(z)))$
 \downarrow
 D : output probabilities
 \rightarrow in $[0, 1]$
 \downarrow
 $D(G(z)) \rightarrow 1$

Generative adversarial networks



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



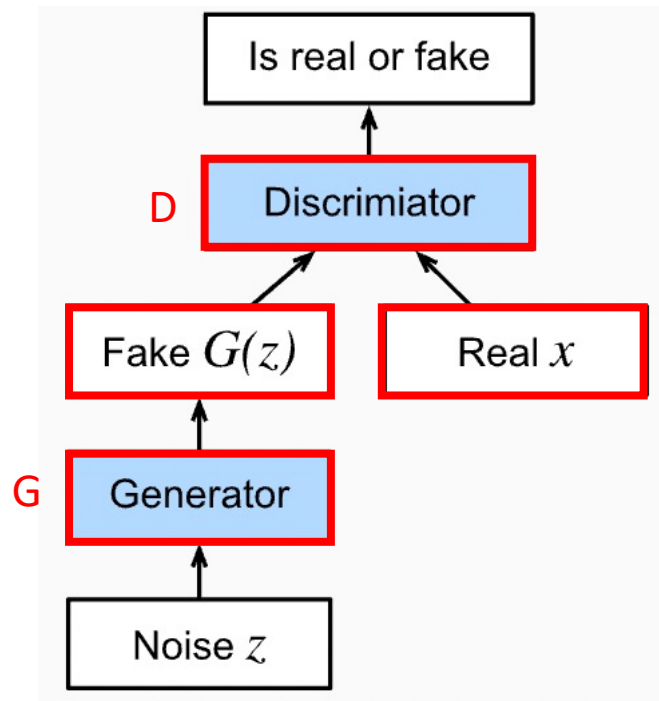
w.r.t. G :

$$\min \log(1 - D(G(z)))$$

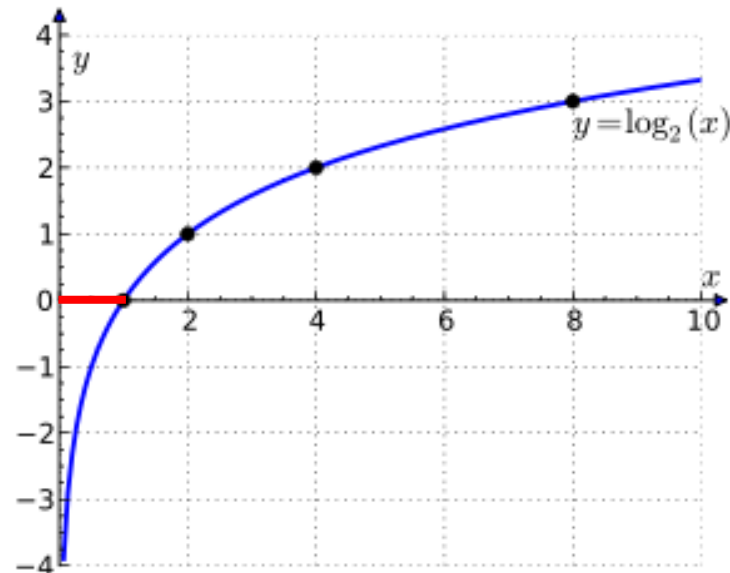
D : output probabilities
 \rightarrow in $[0, 1]$

$$D(G(z)) \rightarrow 1$$

Generative adversarial networks

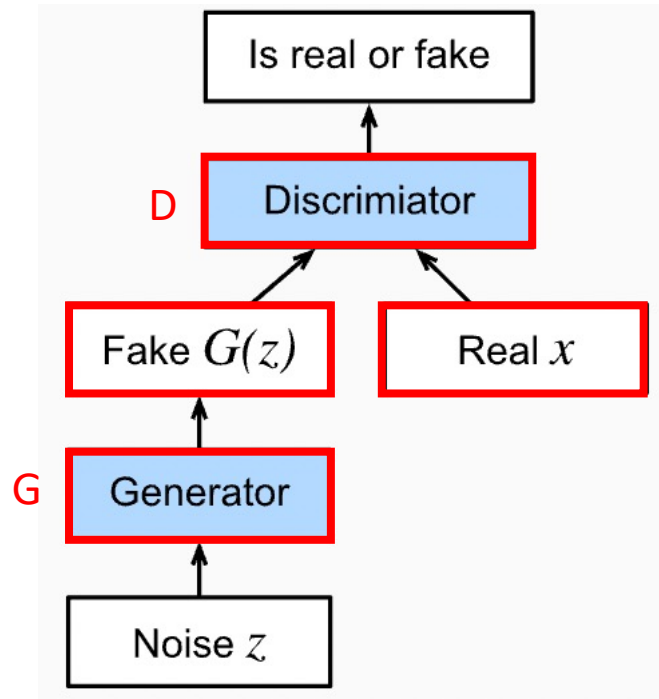


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

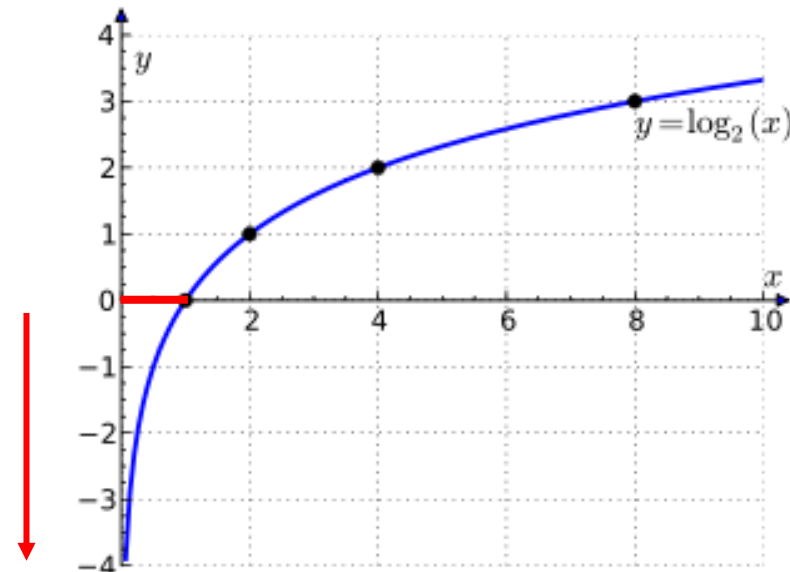


w.r.t. G :
 $\min \log(1 - D(G(z)))$
 \downarrow
 D : output probabilities
 \rightarrow in $[0, 1]$
 \downarrow
 $D(G(z)) \rightarrow 1$

Generative adversarial networks



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



w.r.t. G :

$$\min \log(1 - D(G(z)))$$

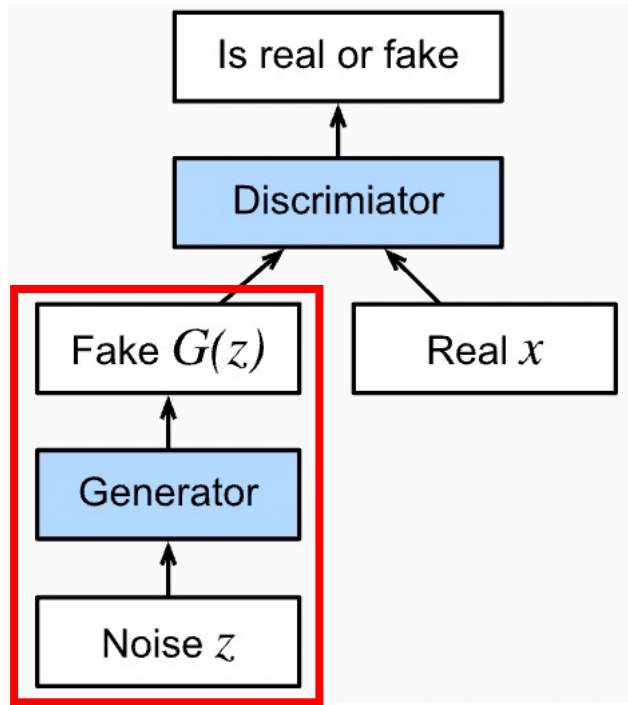
↓

D : output probabilities
→ in $[0, 1]$

↓

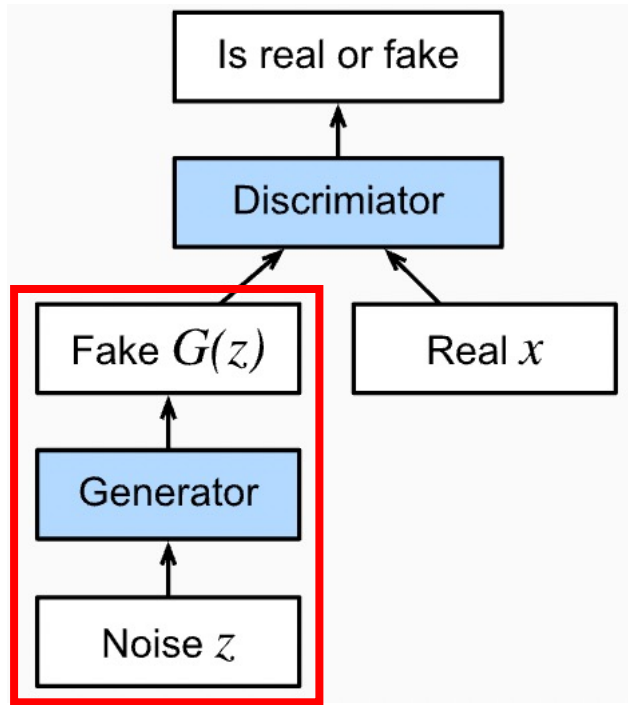
$D(G(z)) \rightarrow 1$

GAN for image data

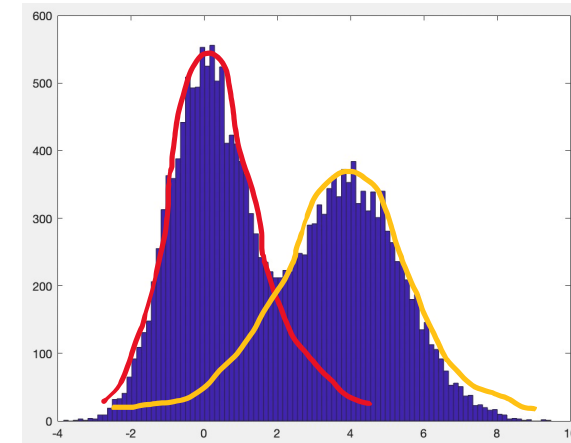


Q: how to generate data?

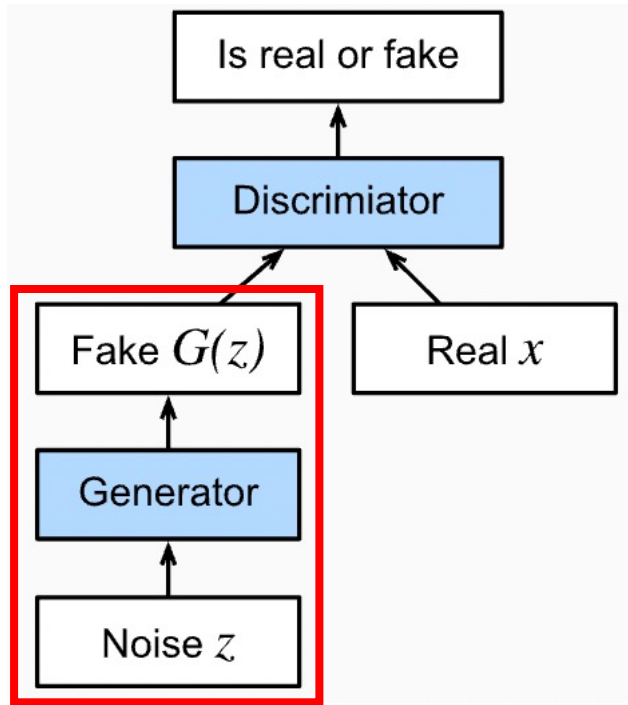
GAN for image data



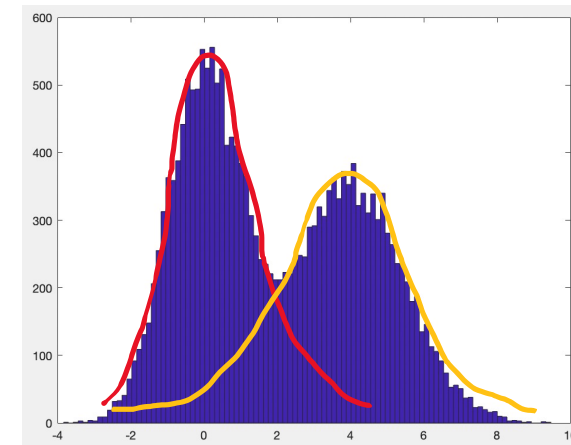
Q: how to generate data?



GAN for image data

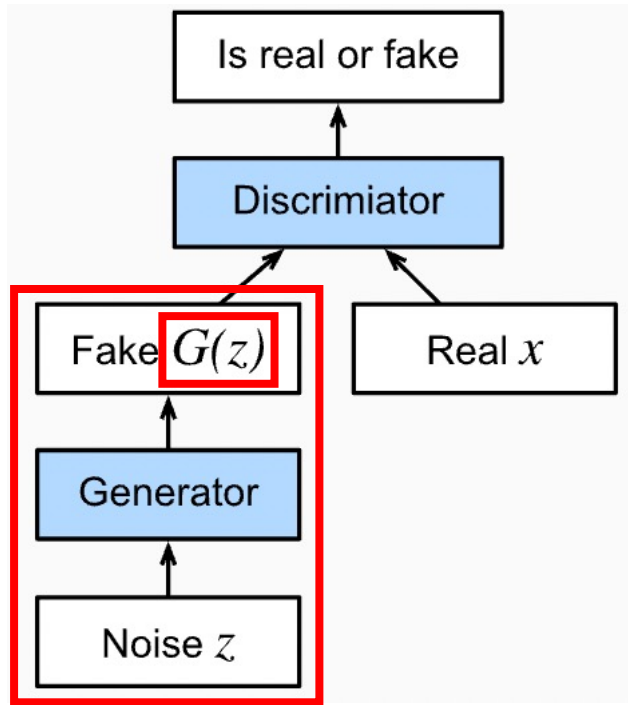


Q: how to generate data?

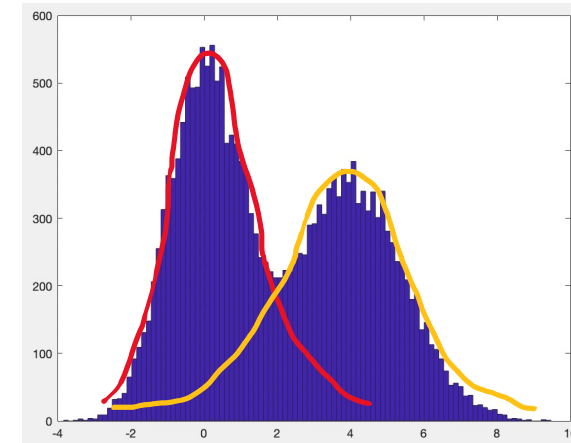


Explicitly model the data distribution

GAN for image data

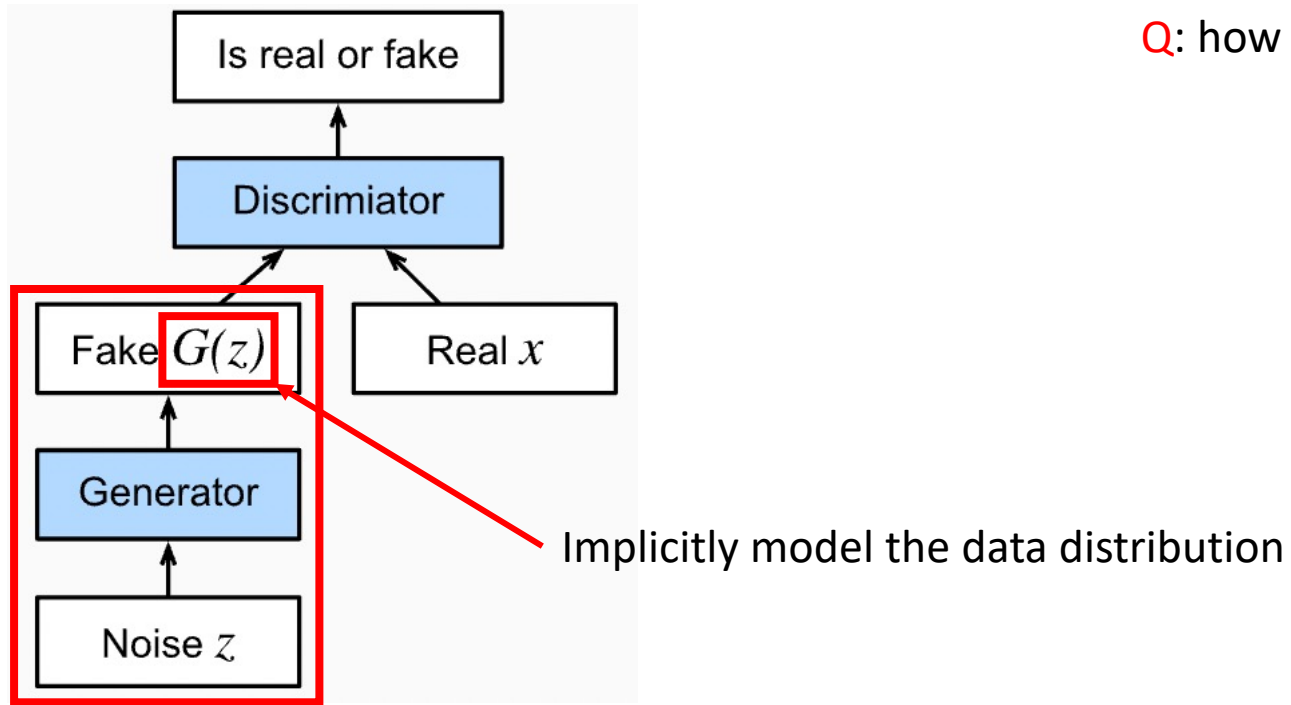


Q: how to generate data?

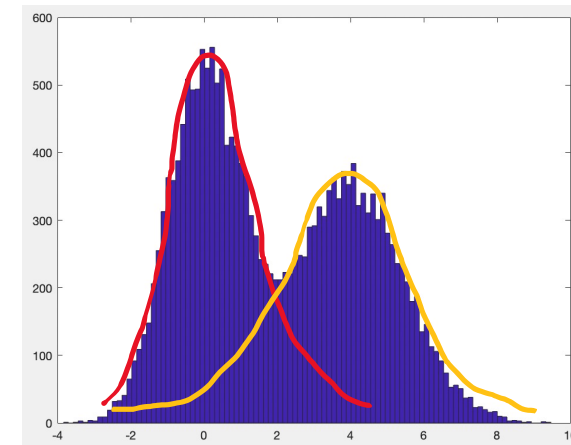


Explicitly model the data distribution

GAN for image data

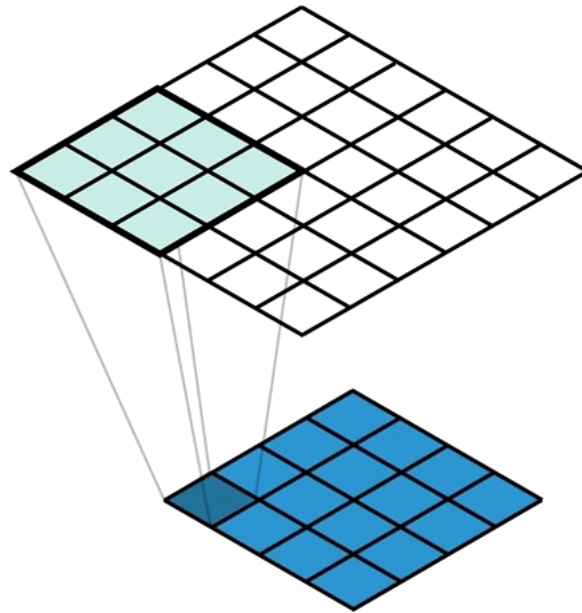


Q: how to generate data?



Explicitly model the data distribution

Transposed convolution



Transposed convolution

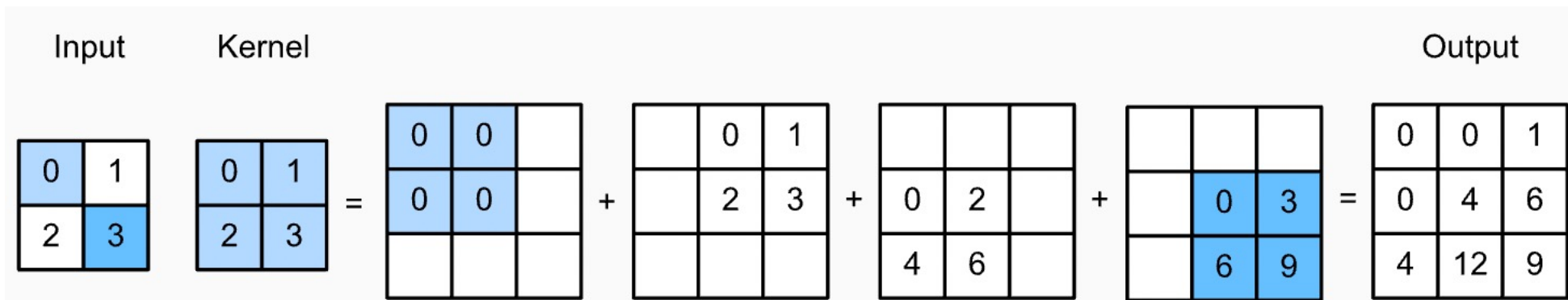
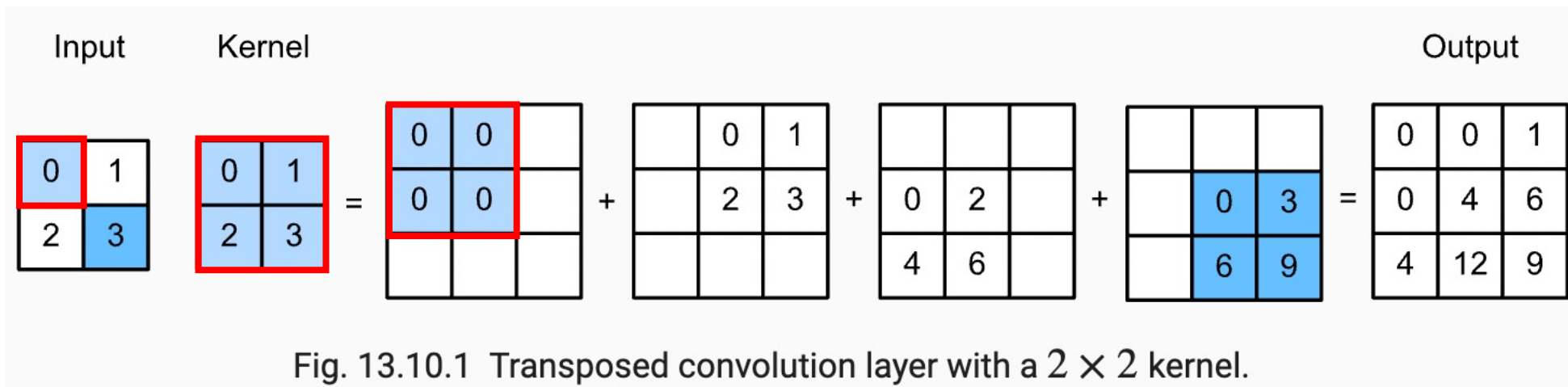
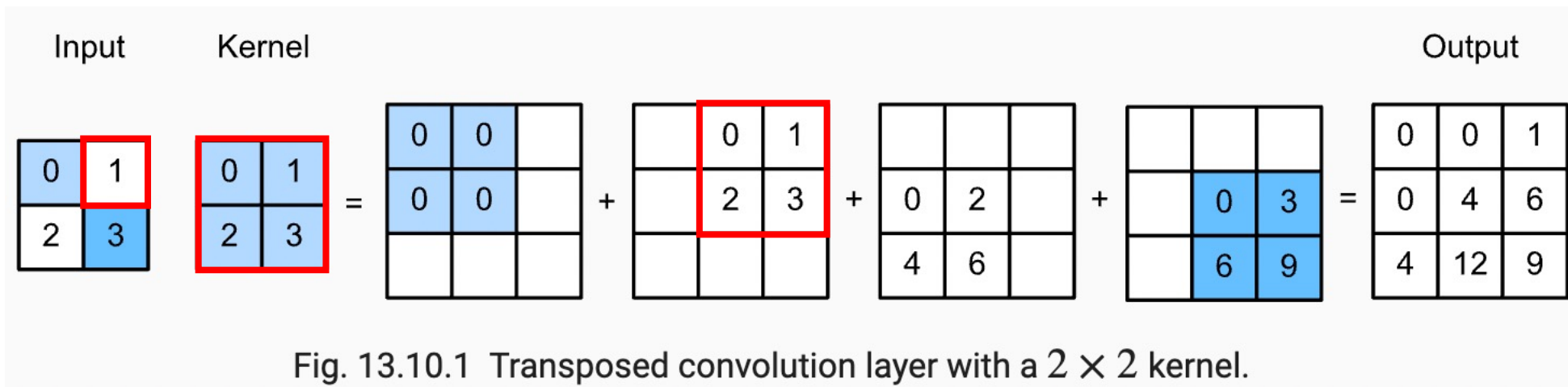


Fig. 13.10.1 Transposed convolution layer with a 2×2 kernel.

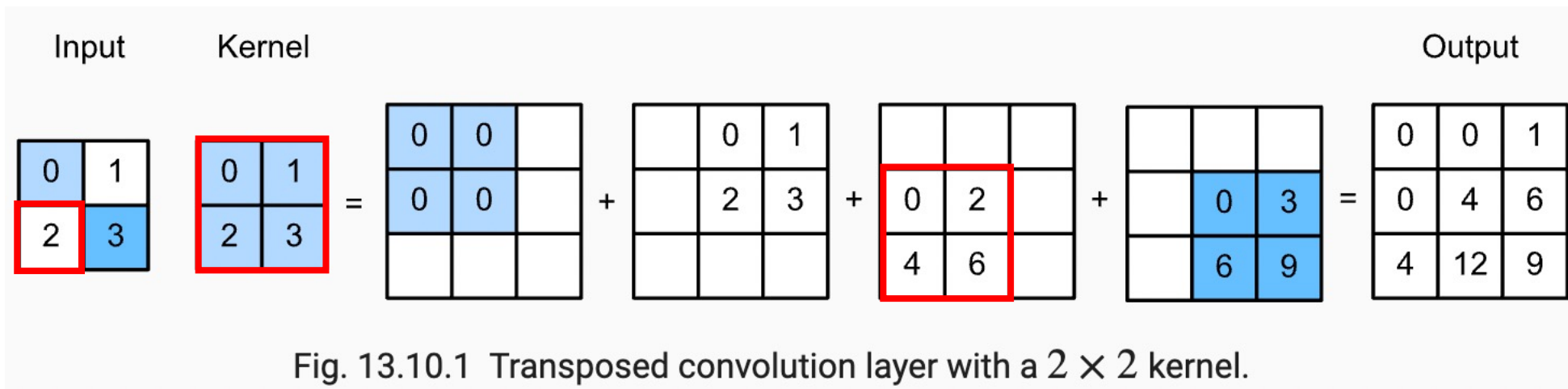
Transposed convolution



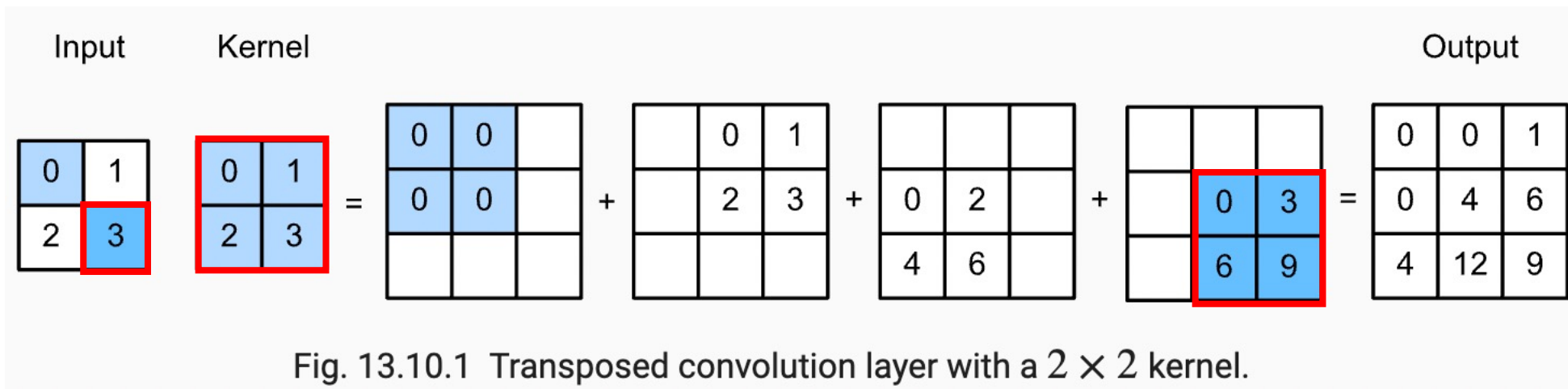
Transposed convolution



Transposed convolution



Transposed convolution



Transposed convolution

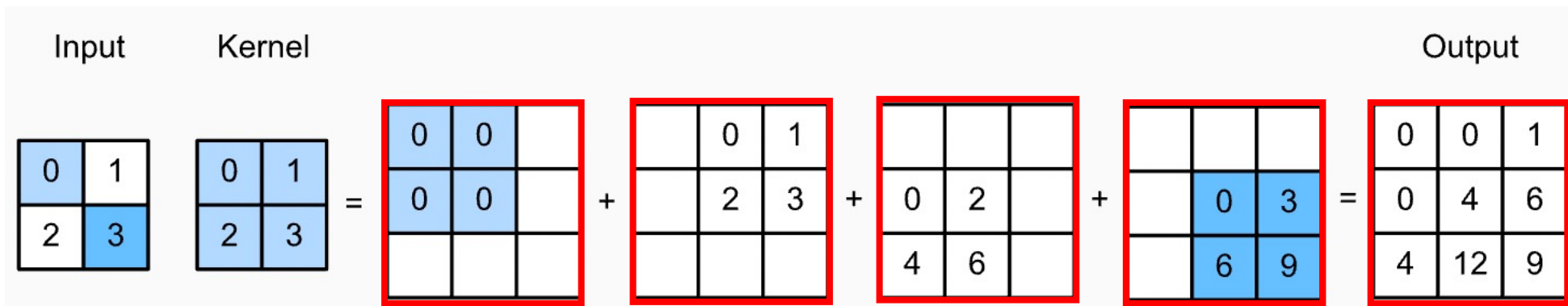
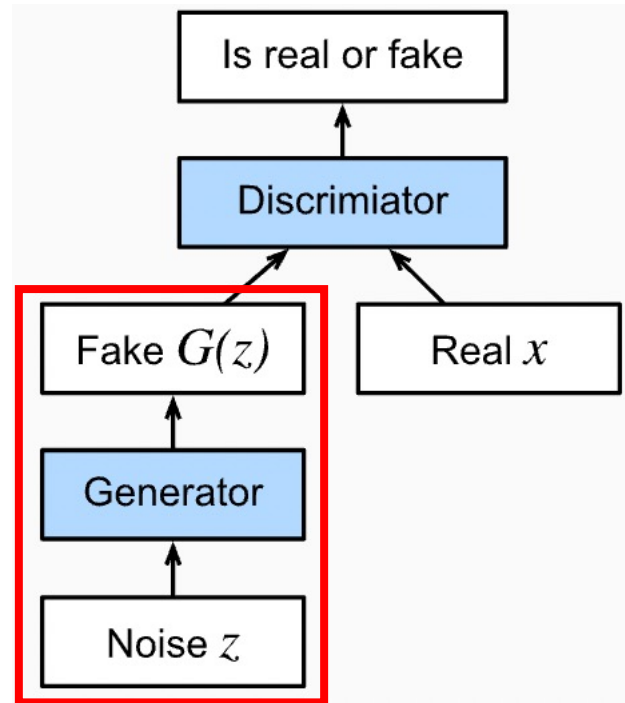


Fig. 13.10.1 Transposed convolution layer with a 2×2 kernel.

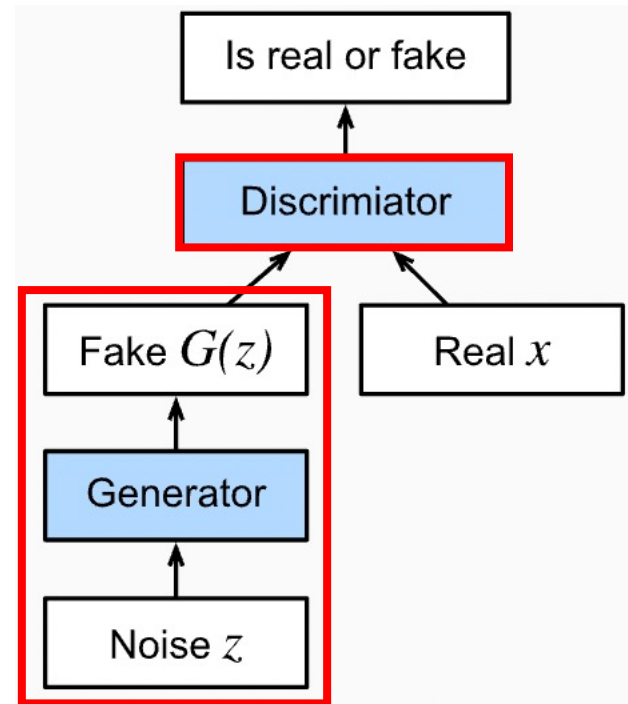
GAN for image data

Transposed convolution



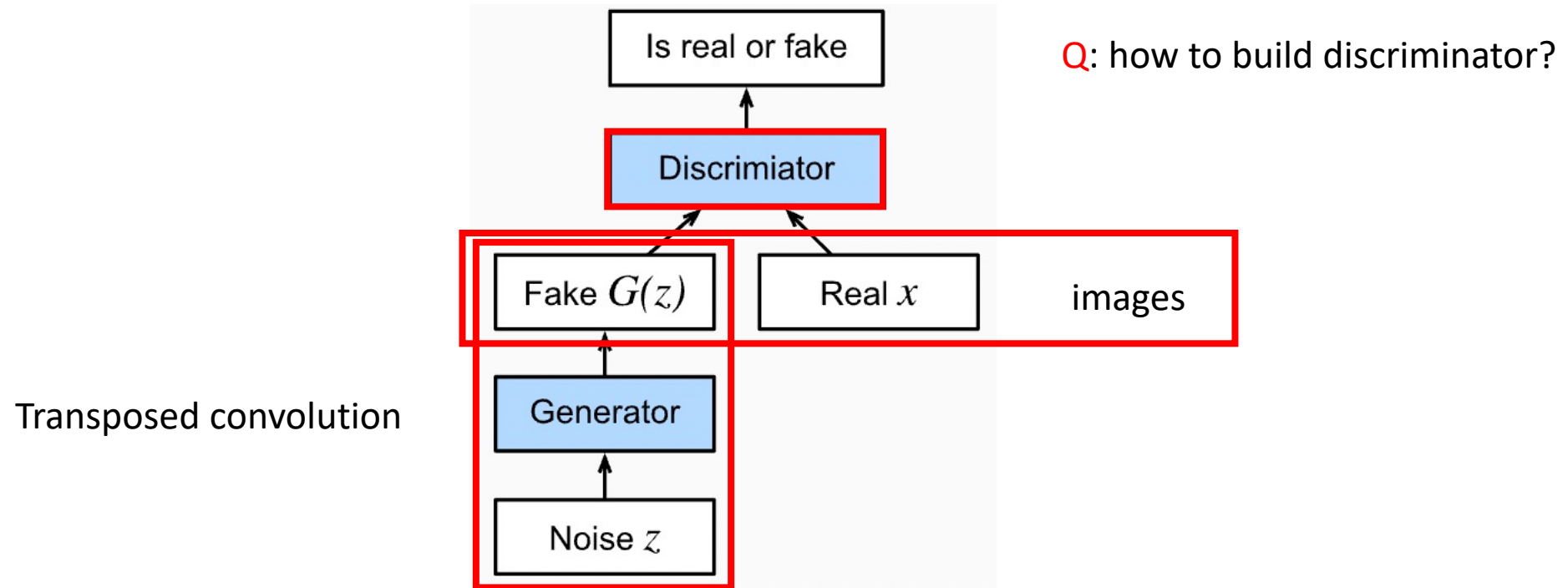
GAN for image data

Transposed convolution

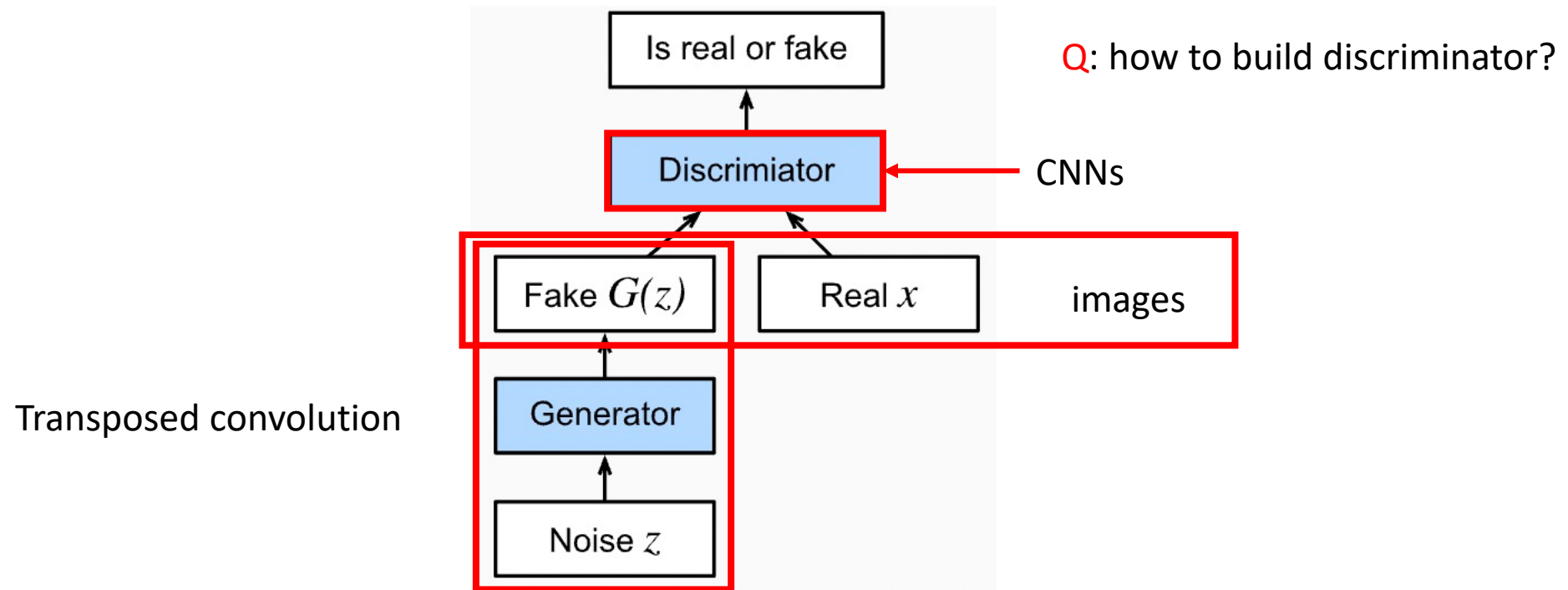


Q: how to build discriminator?

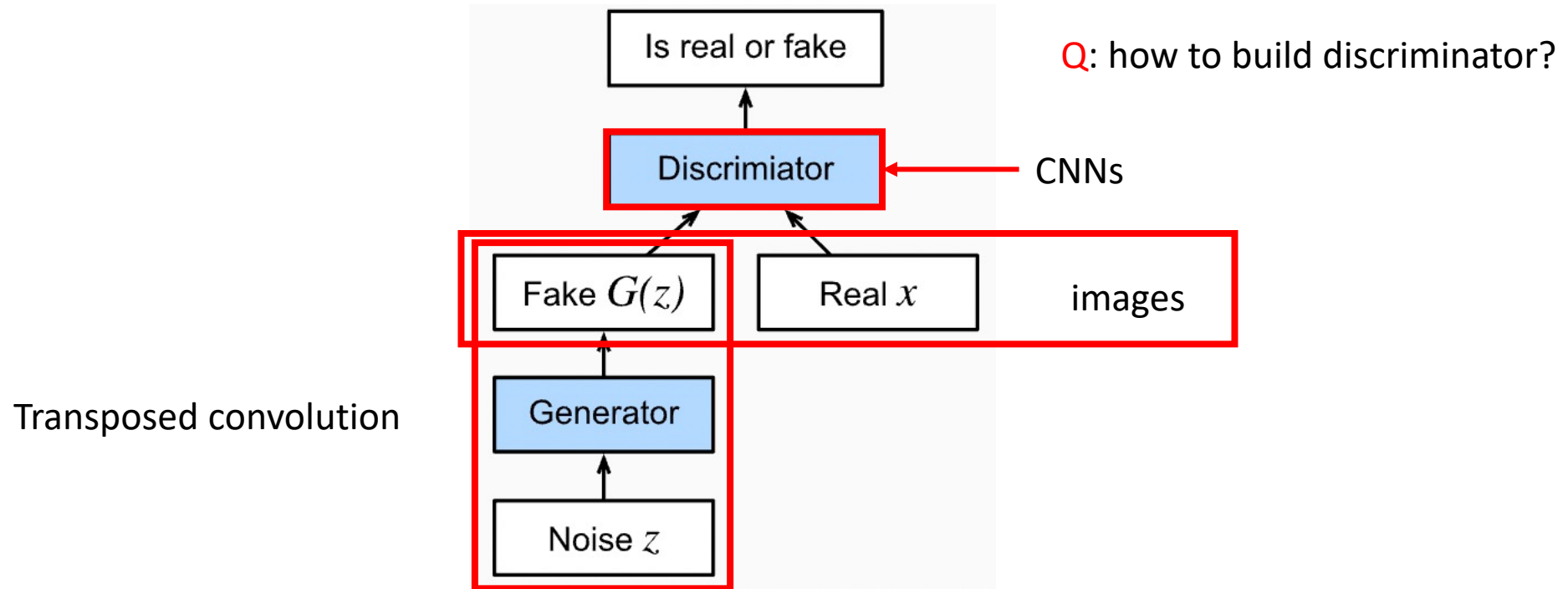
GAN for image data



GAN for image data



GAN for image data



Q: how to optimize the two elements (D and G)?

Optimization of GAN

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

Optimization of GAN

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

Optimization of GAN

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

Optimization of GAN

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

Optimization of GAN

Algorithm 1

Steps to apply to experiments.

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

Optimization of GAN

Algorithm 1 Mini-batch gradient descent steps to apply to GAN experiments.

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)}))) \right] .$$
 Stochastic gradients w.r.t. **discriminator** parameters

end for

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

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

end for

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

Optimization of GAN

Algorithm 1 Minimax steps to apply to experiments.

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))]$$
 Stochastic gradients w.r.t. discriminator parameters

end for

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

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(\mathbf{z}^{(i)})))$$
 Gradients w.r.t. generator parameters

end for

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

Optimization of GAN

Algorithm 1 M
steps to apply to
experiments.

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

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

BUT: training GAN is difficult

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))]$$
 Stochastic gradients w.r.t. **discriminator** parameters

end for

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

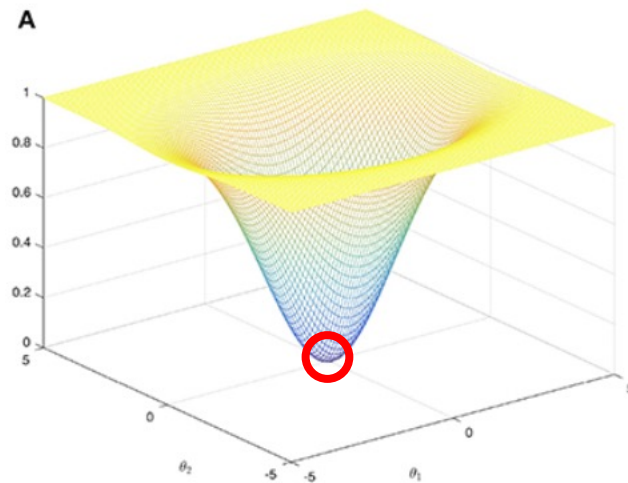
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(\mathbf{z}^{(i)})))$$
 Gradients w.r.t. generator parameters

end for

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

Difficulty of training GANs

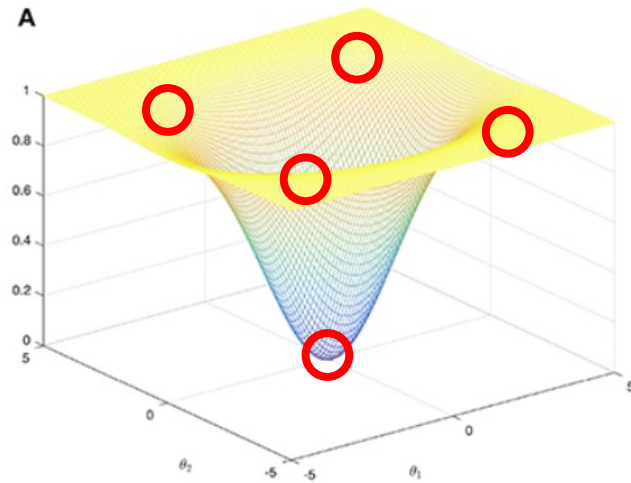
- Difficult to converge



A convex objective function

Difficulty of training GANs

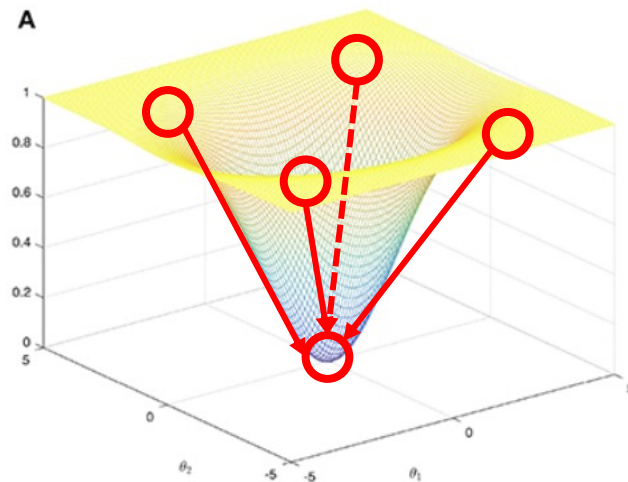
- Difficult to converge



A convex objective function

Difficulty of training GANs

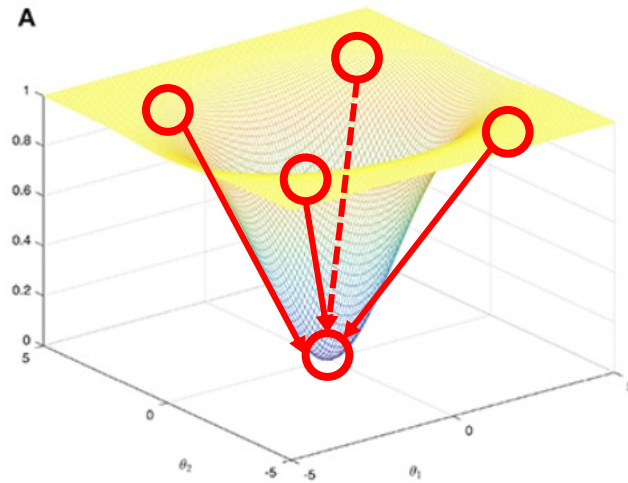
- Difficult to converge



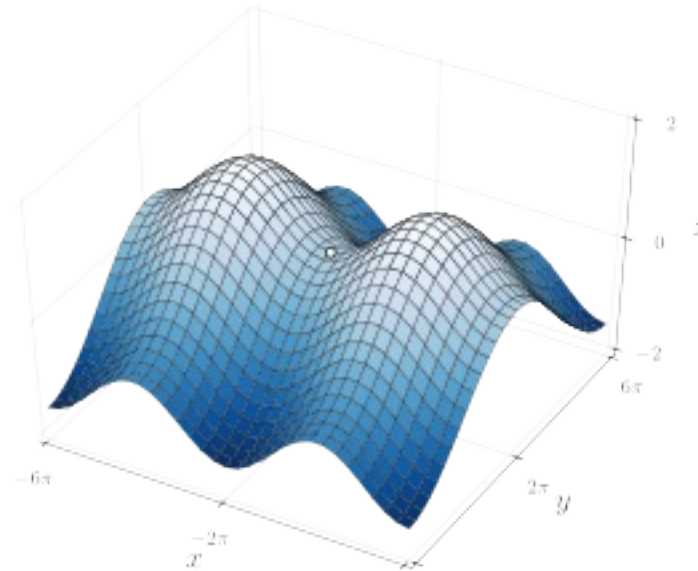
A convex objective function

Difficulty of training GANs

- Difficult to converge



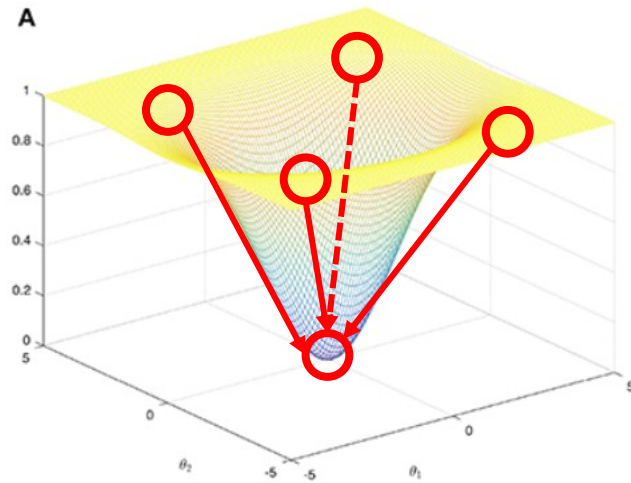
A convex objective function



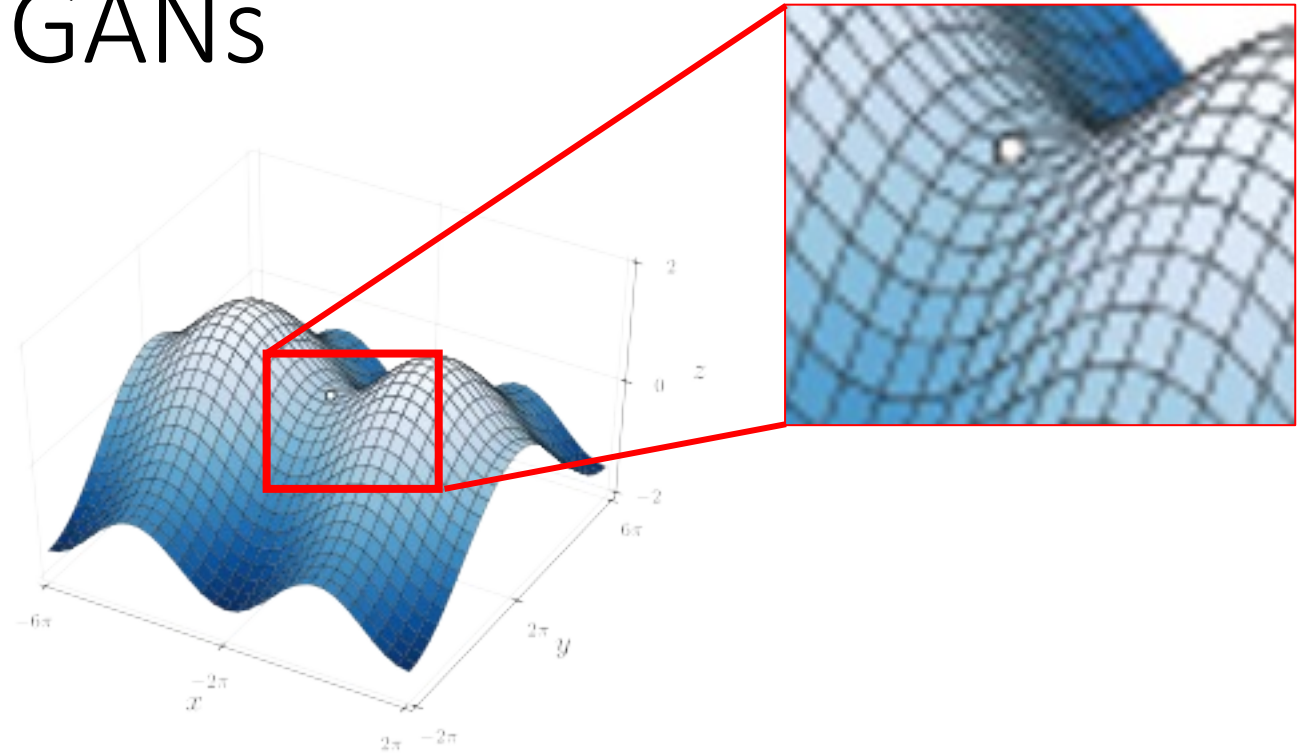
Nonconvex function

Difficulty of training GANs

- Difficult to converge

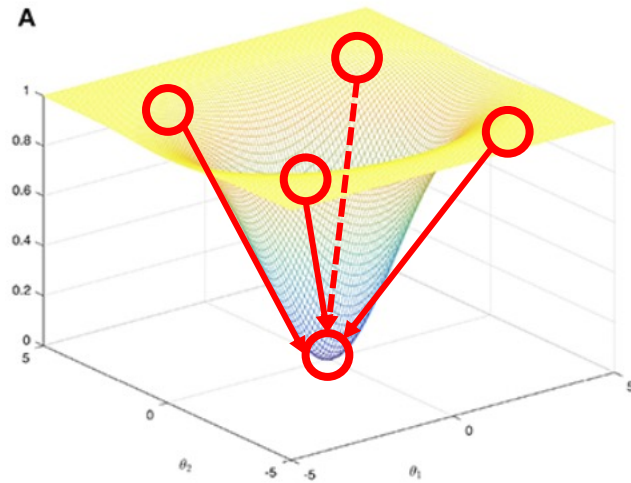


A convex objective function

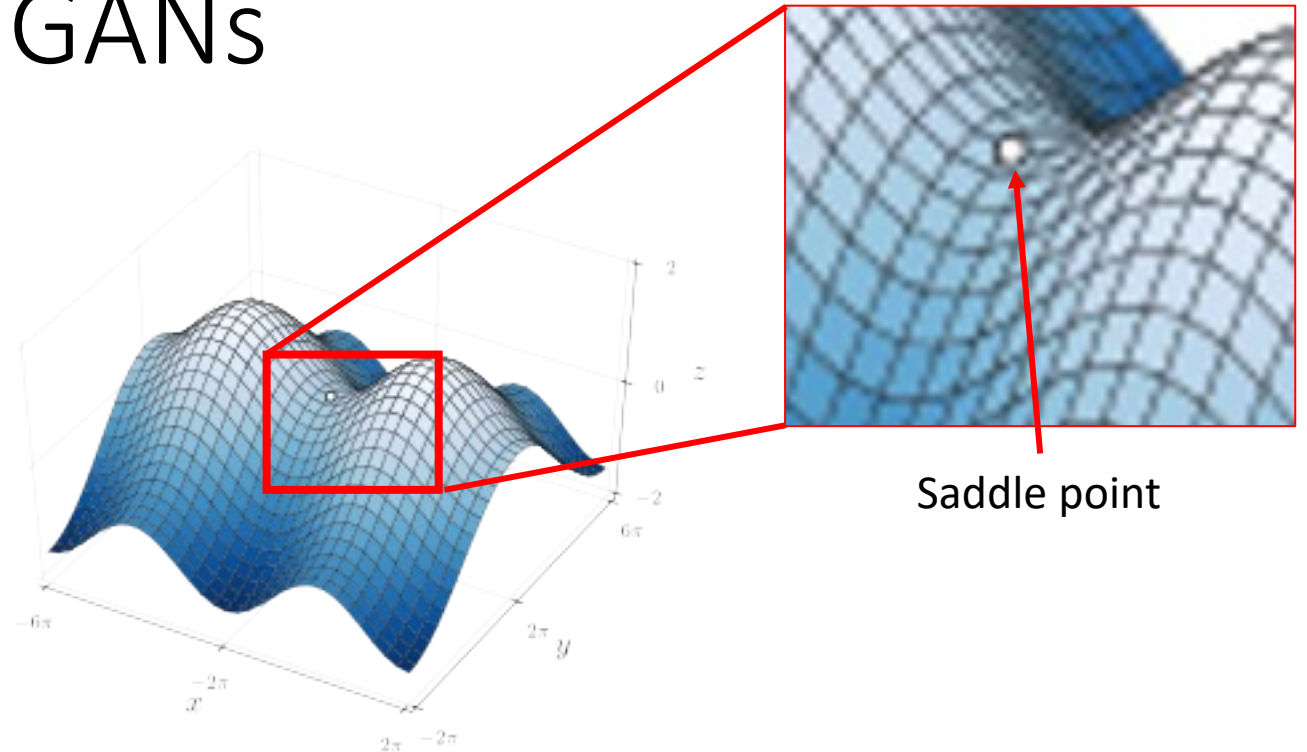


Difficulty of training GANs

- Difficult to converge



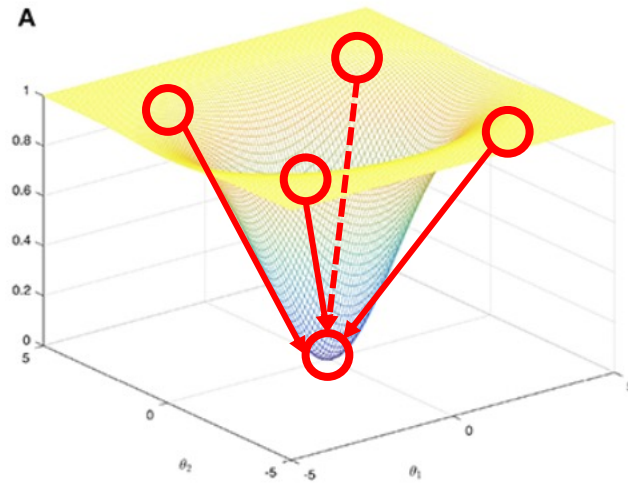
A convex objective function



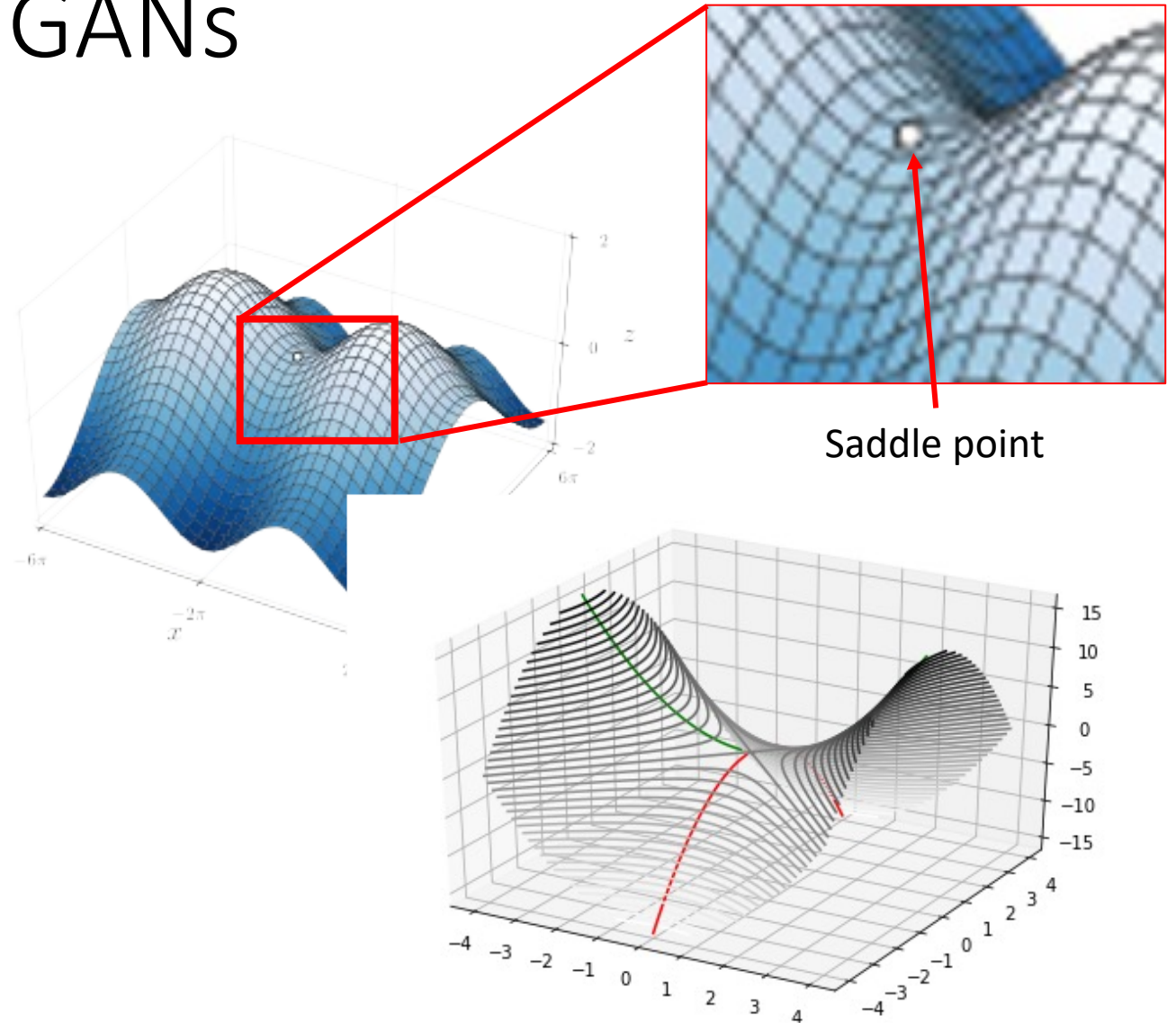
Saddle point

Difficulty of training GANs

- Difficult to converge

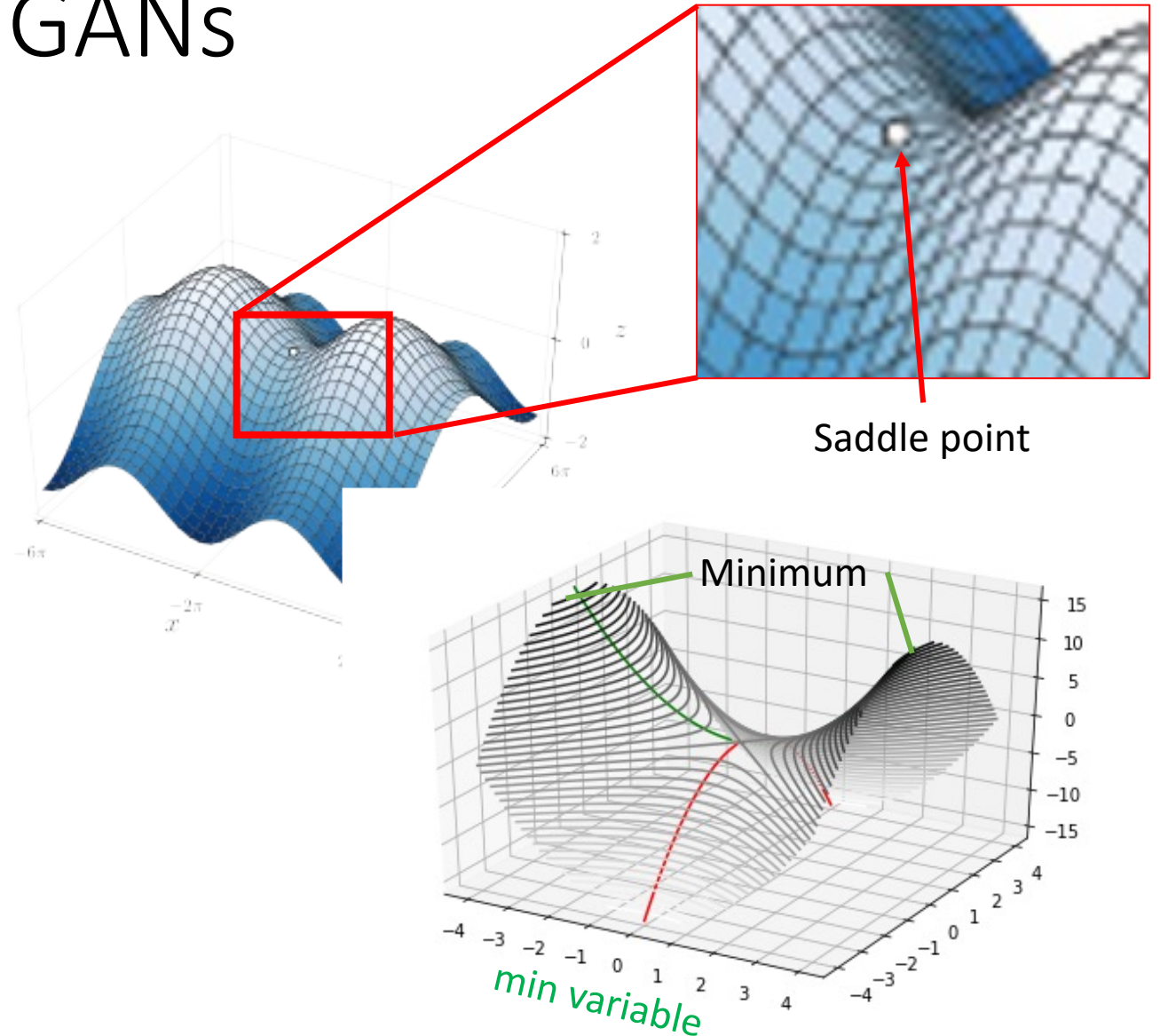
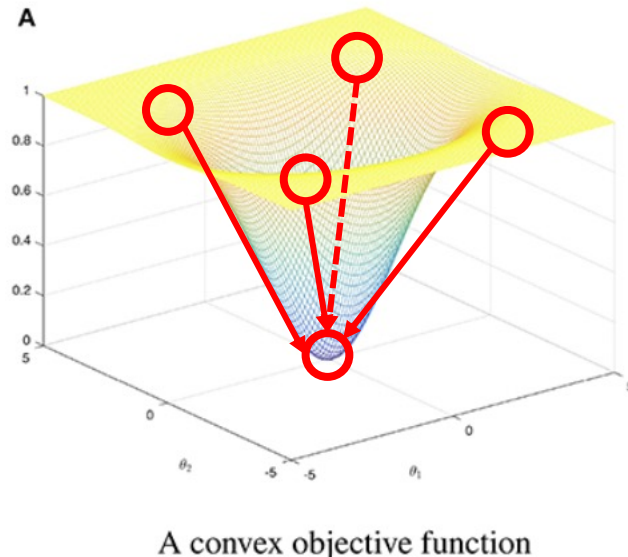


A convex objective function



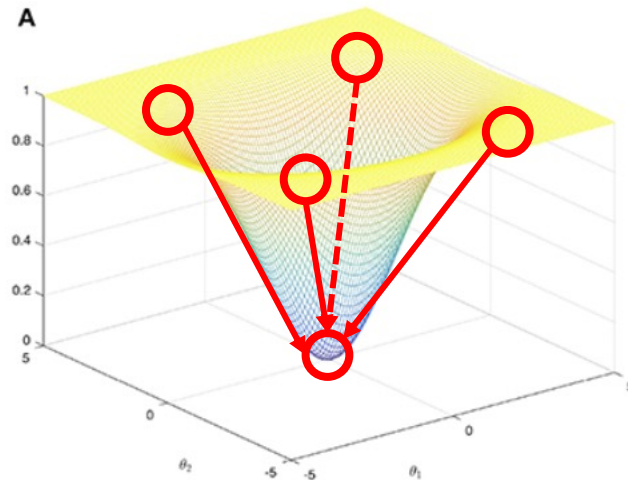
Difficulty of training GANs

- Difficult to converge

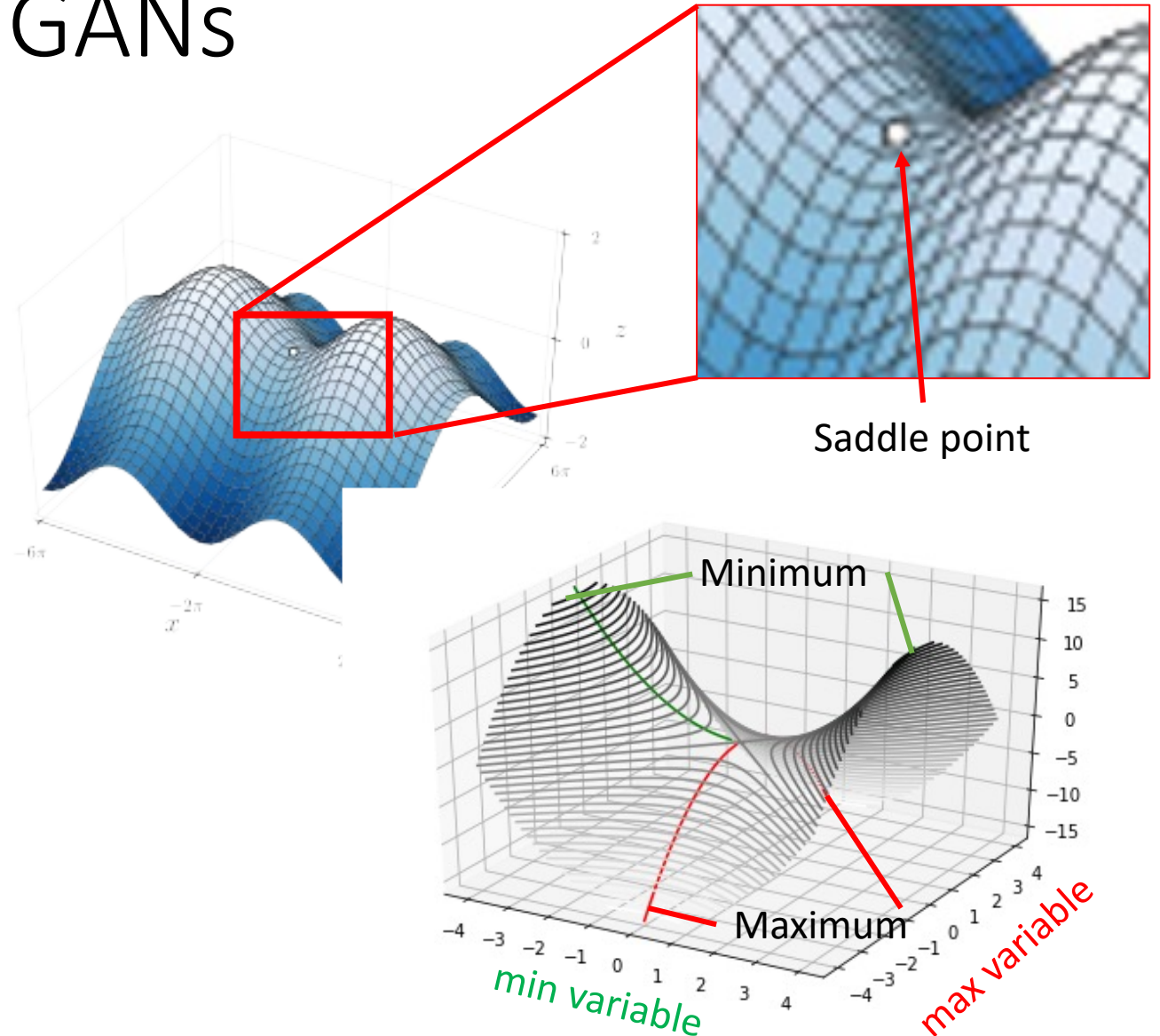


Difficulty of training GANs

- Difficult to converge

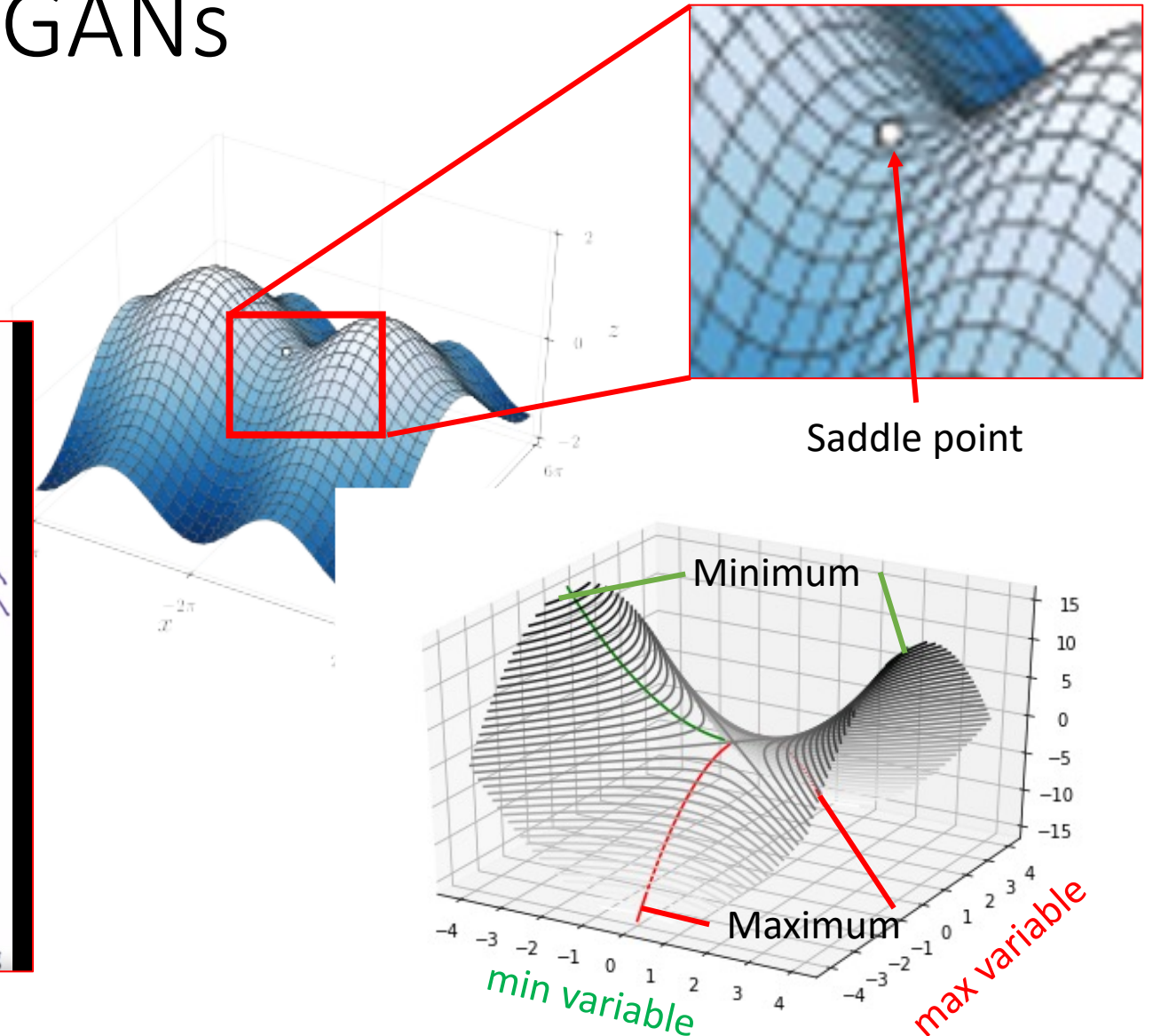
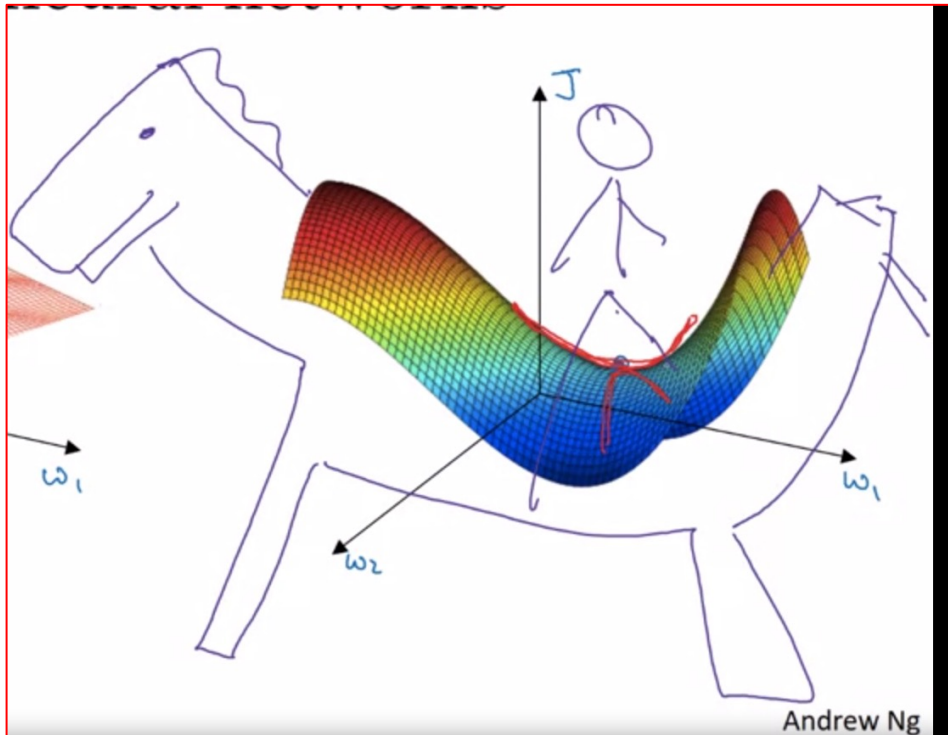


A convex objective function



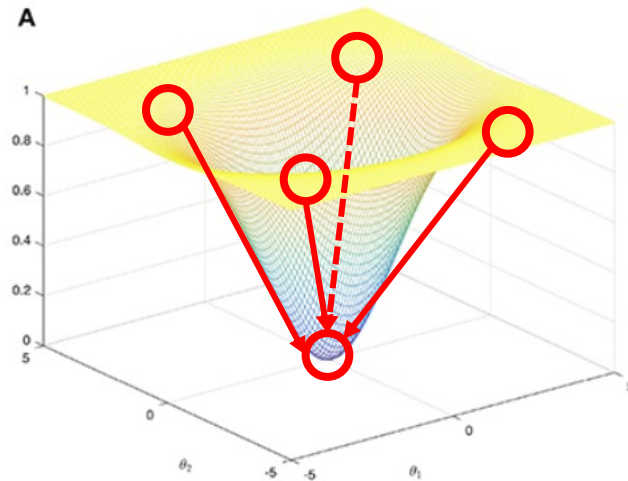
Difficulty of training GANs

- Difficult to converge

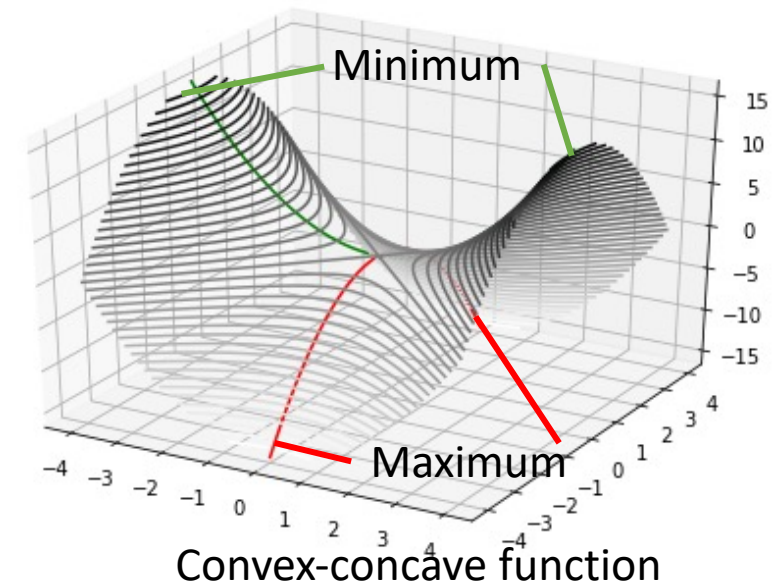
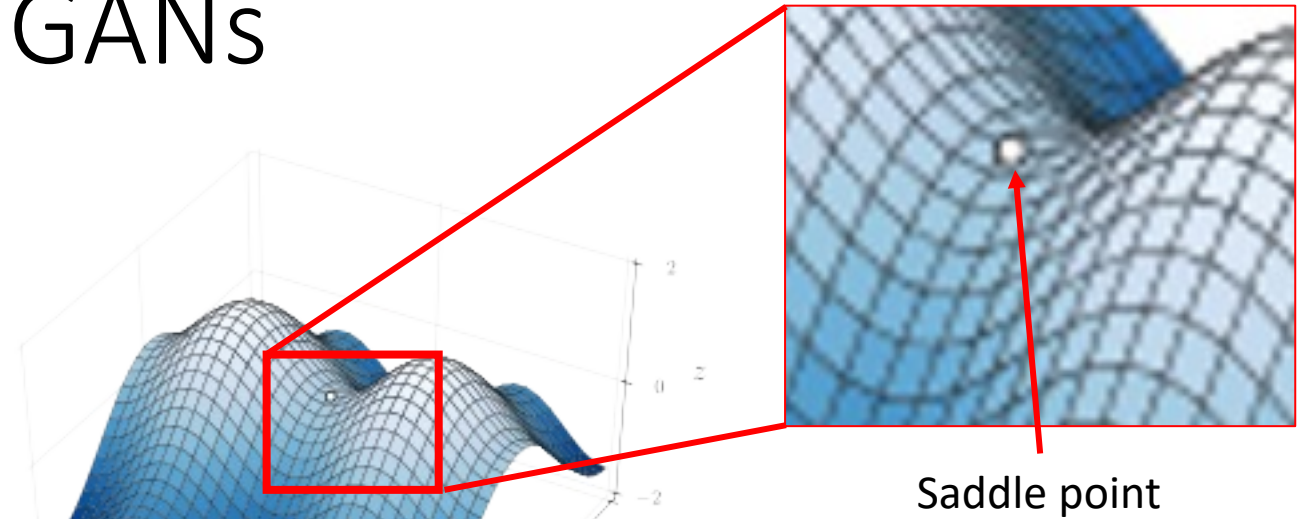


Difficulty of training GANs

- Difficult to converge



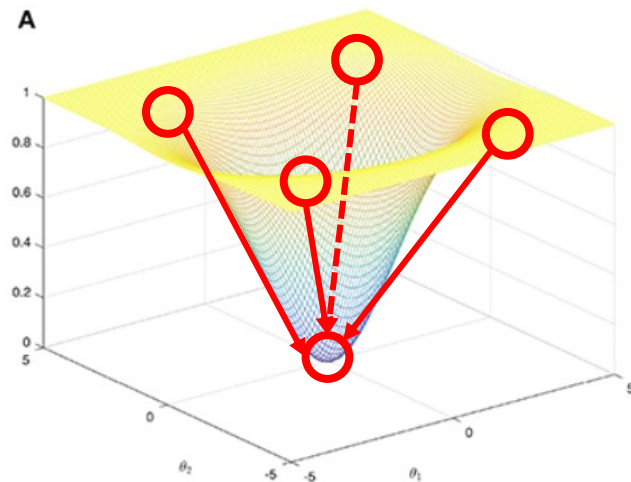
A convex objective function



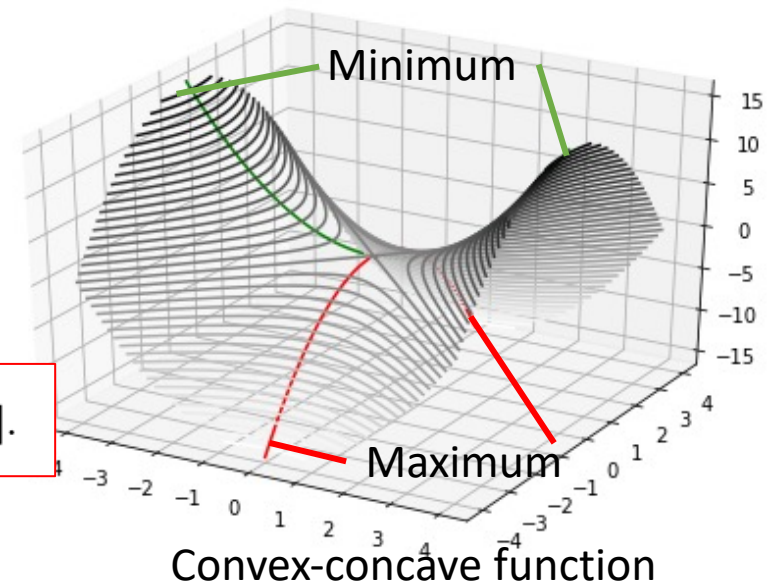
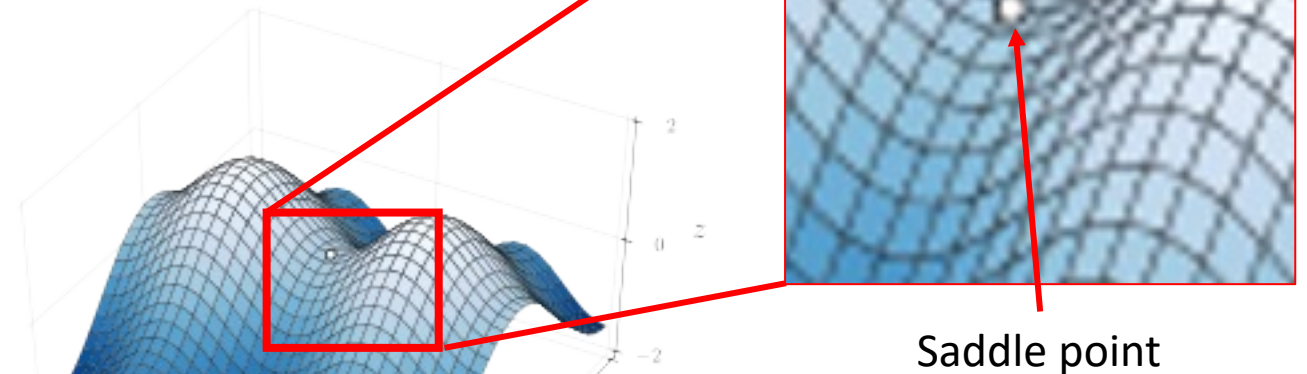
Convex-concave function

Difficulty of training GANs

- Difficult to converge

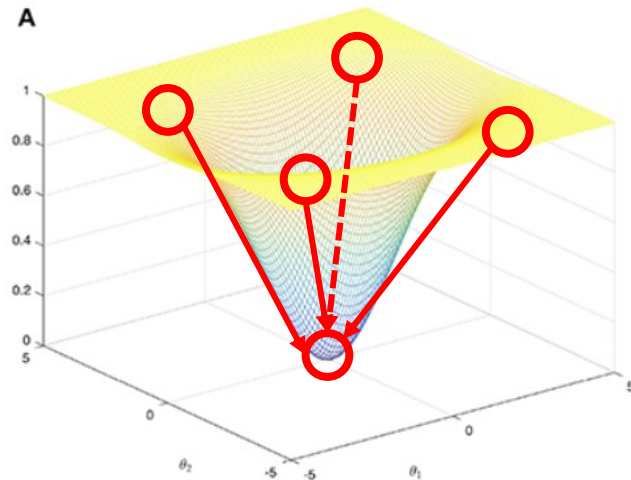


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$



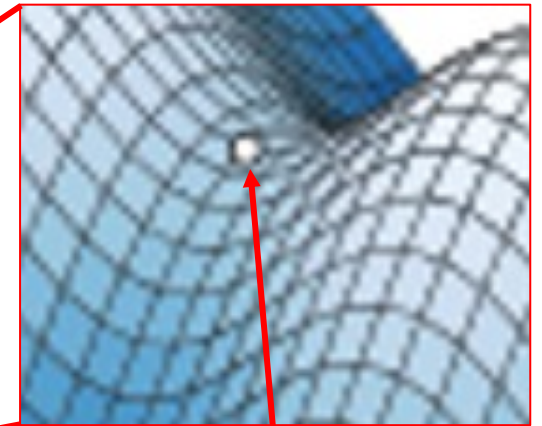
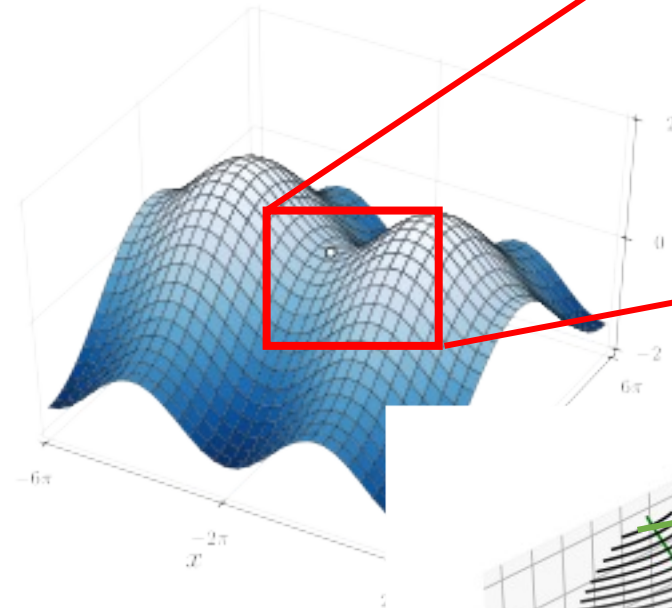
Difficulty of training GANs

- Difficult to converge

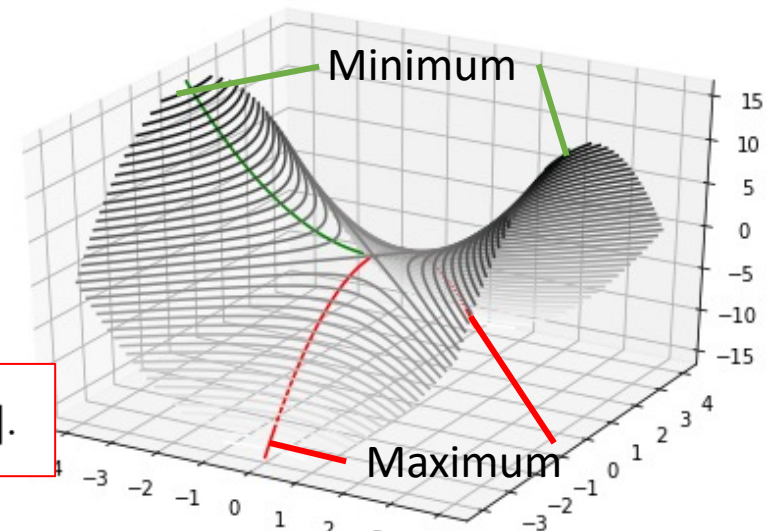


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

nonconvex-nonconcave function



Saddle point



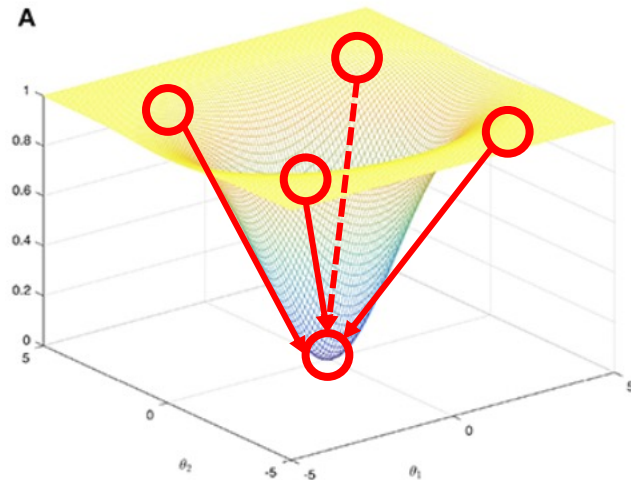
Minimum

Maximum

Convex-concave function

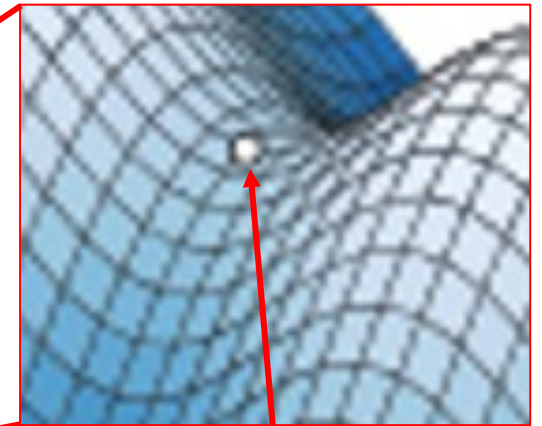
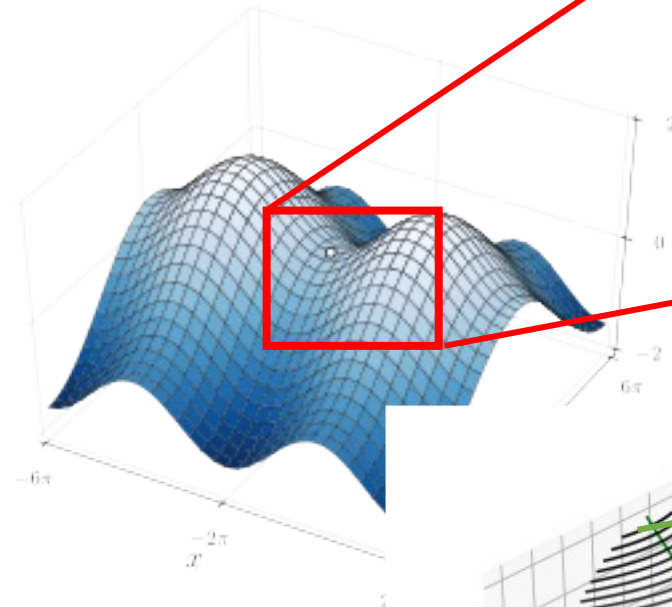
Difficulty of training GANs

- Difficult to converge

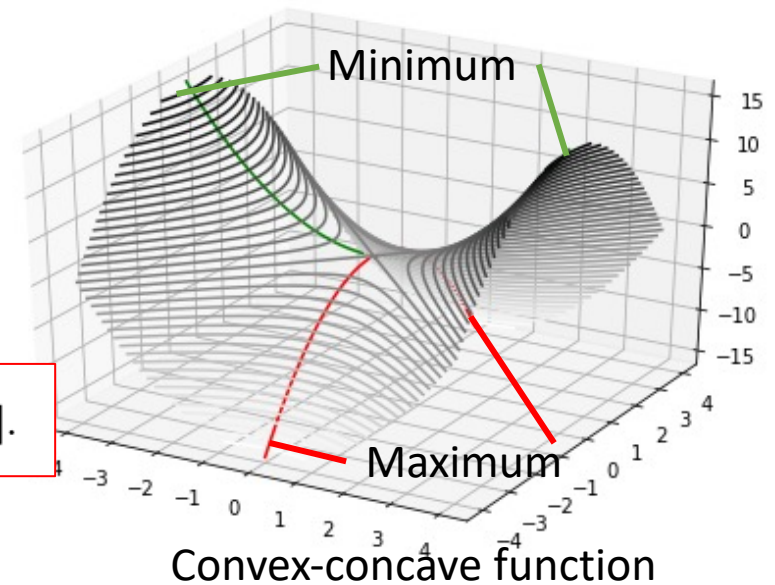


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

nonconvex-nonconcave function
(Hard to define what convergence could be)



Saddle point



Convex-concave function

Recall: convergence rate of optimizers


- When to terminate GD (determining T)?
 - How to measure the approximation error?

$$f(w_T) - f(w_*) \leq \epsilon$$

Recall: convergence rate of optimizers

- When to terminate GD (determining T)?
 - How to measure the approximation error?

$$f(w_T) - f(w_*) \leq \epsilon$$



Level of accuracy:
 ϵ -accurate solution

Recall: convergence rate of optimizers

- When to terminate GD (determining T)?
 - How to measure the approximation error?

$$f(w_T) - f(w_*) \leq \epsilon$$

Level of accuracy:
 ϵ -accurate solution

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

Recall: convergence rate of optimizers

- When to terminate GD (determining T)?
 - How to measure the approximation error?

$$f(w_T) - f(w_*) \leq \epsilon$$

Level of accuracy:
 ϵ -accurate solution

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis

Recall: convergence rate of optimizers

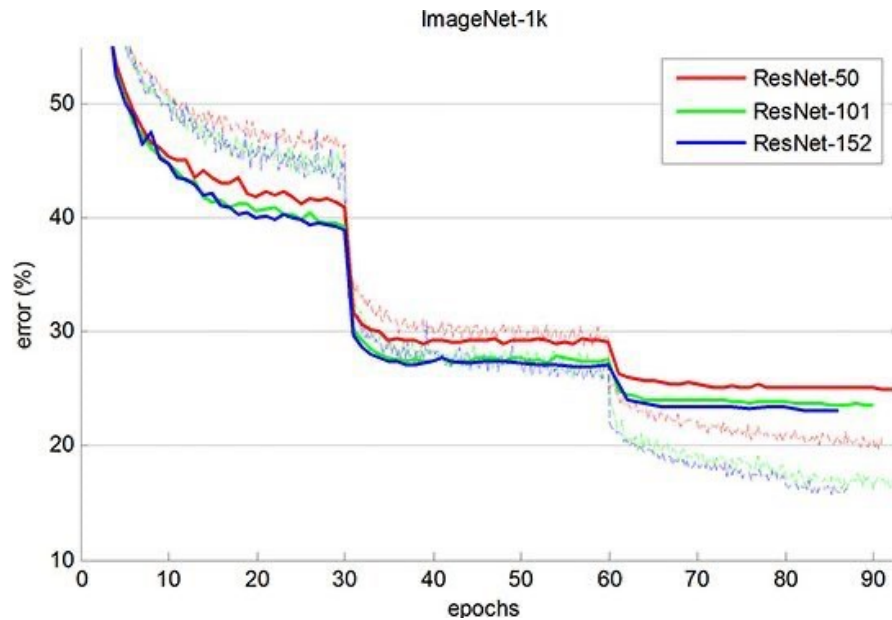
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis



Recall: convergence rate of optimizers

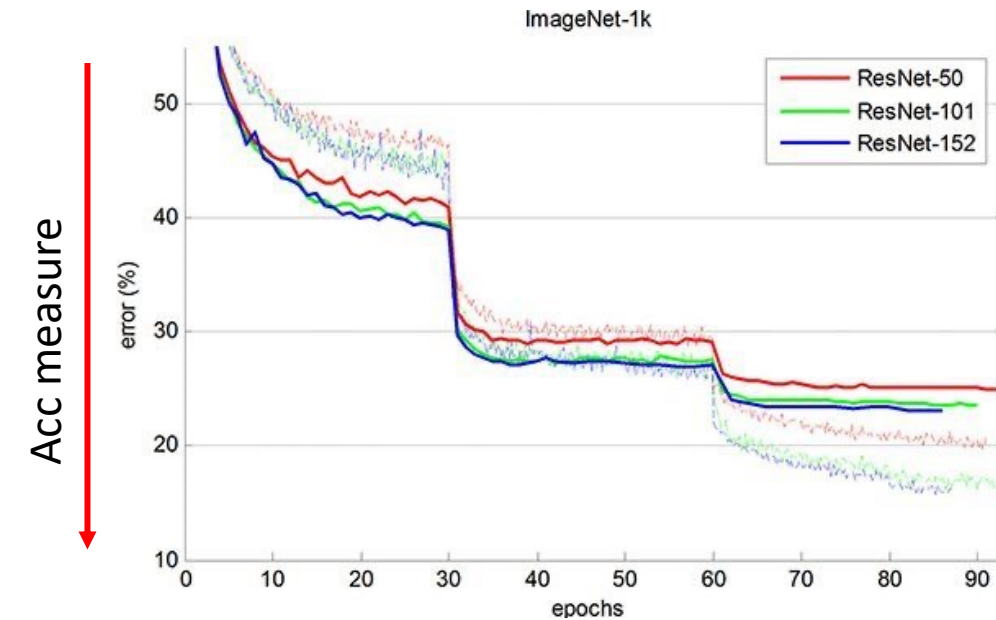
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis



Recall: convergence rate of optimizers

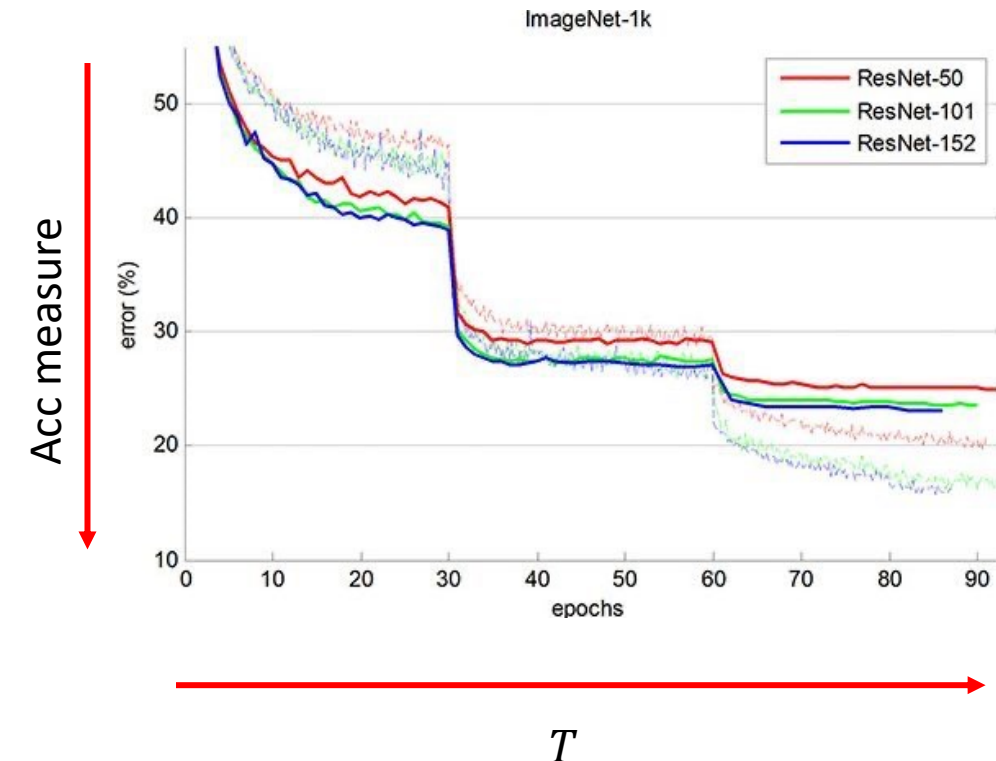
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis



Recall: convergence rate of optimizers

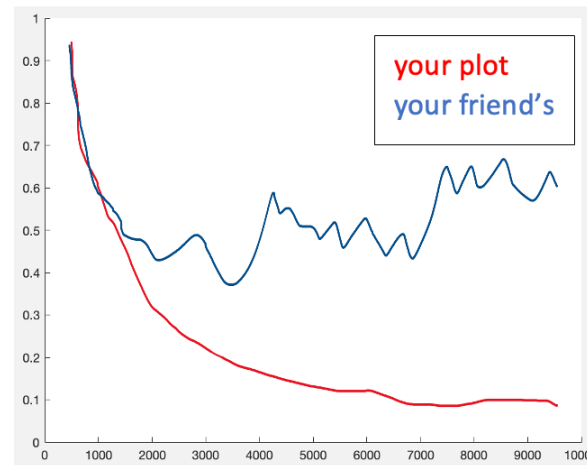
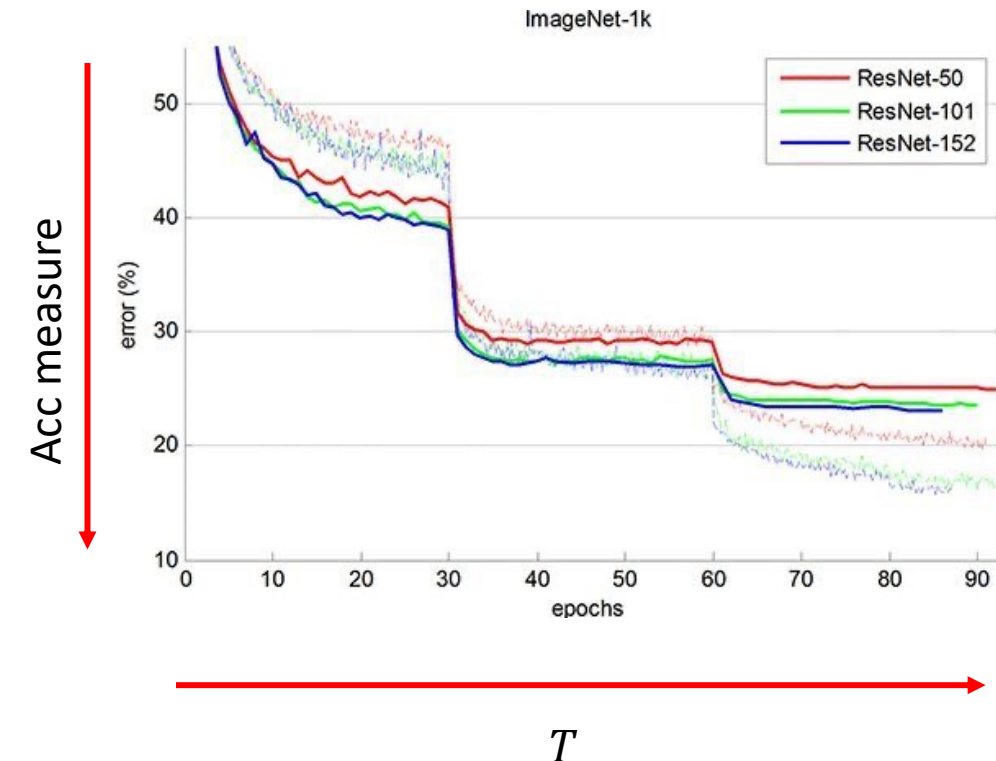
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis



Recall: convergence rate of optimizers

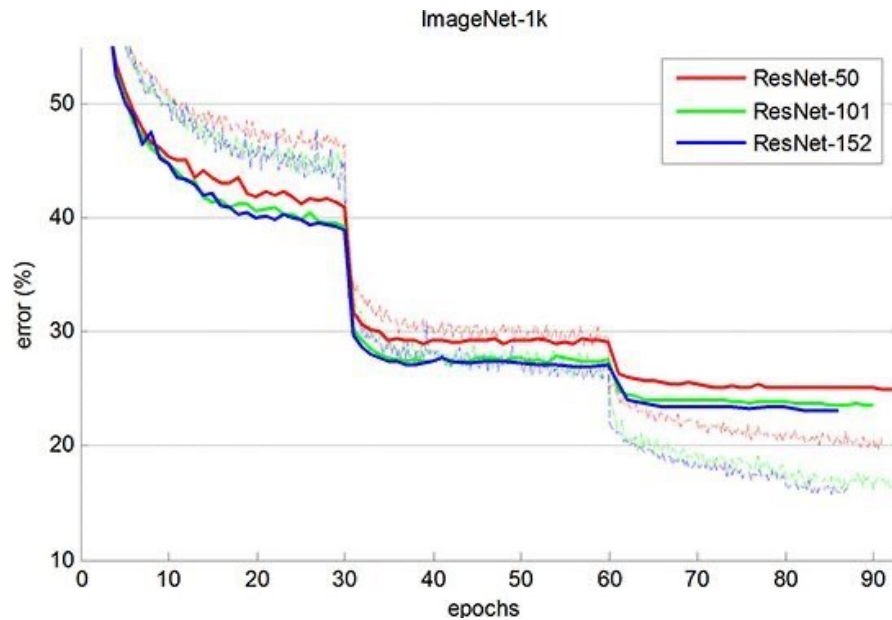
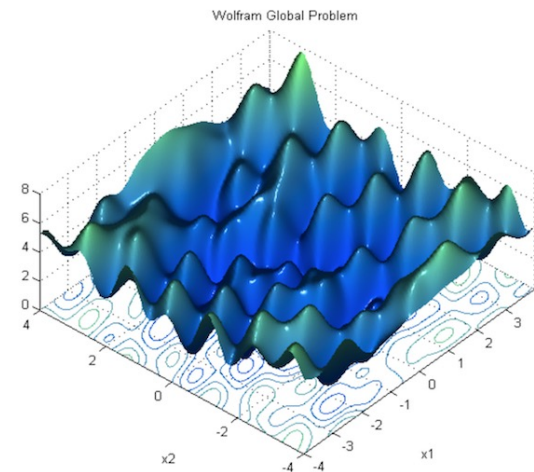
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis



T

Recall: convergence rate of optimizers

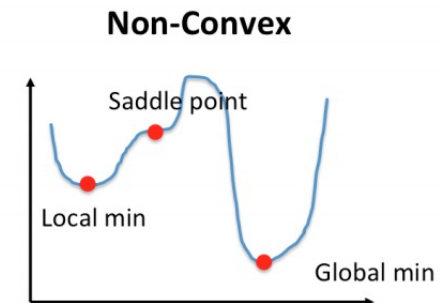
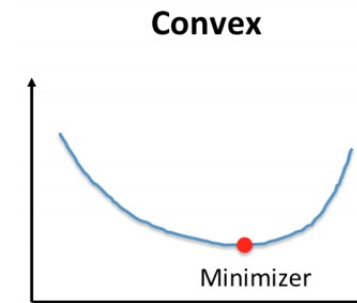
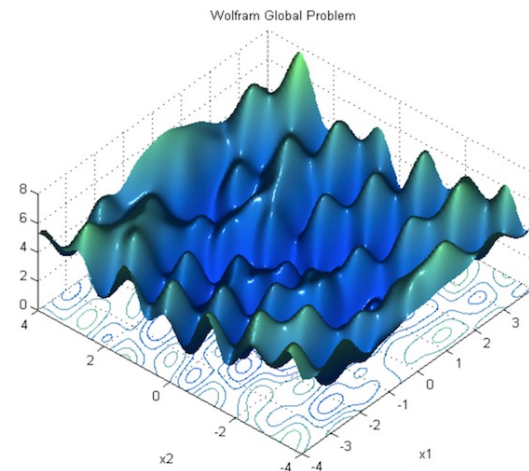
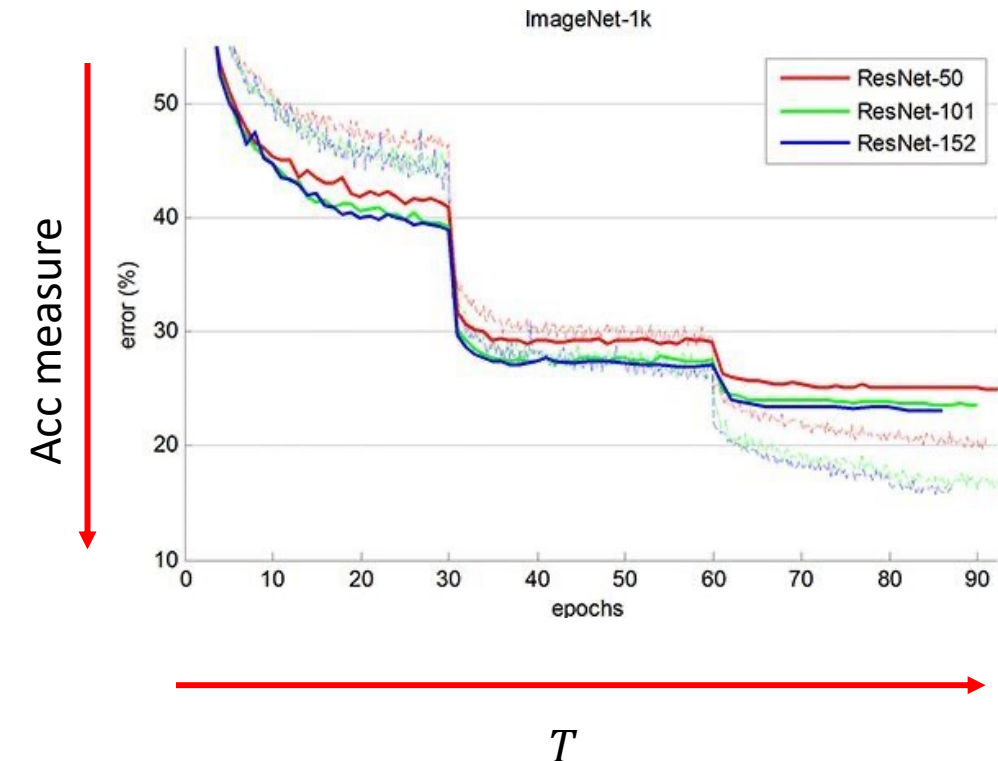
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

No need to know this solution
Only for convergence analysis



Recall: convergence rate of optimizers

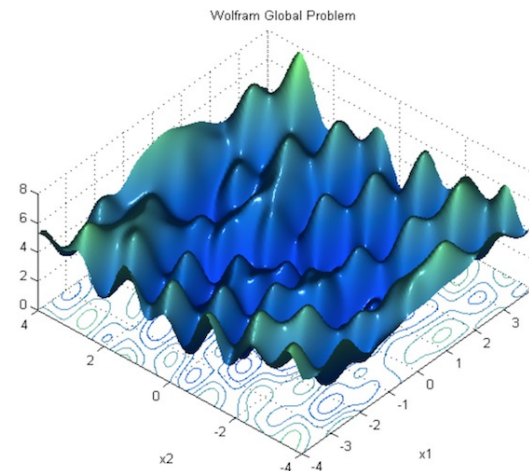
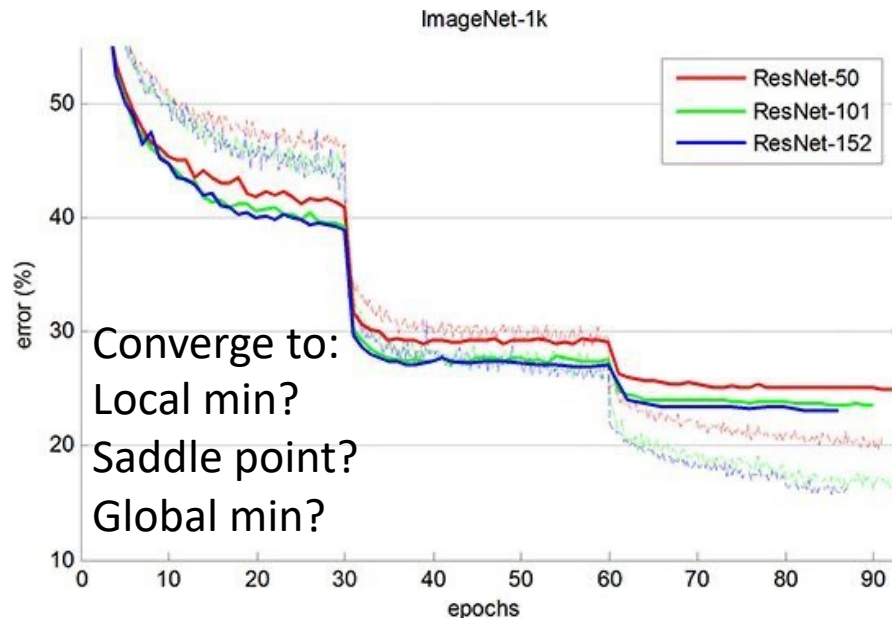
- When to terminate GD (determining T)?
 - How to measure the approximation error?

Level of accuracy:
 ϵ -accurate solution

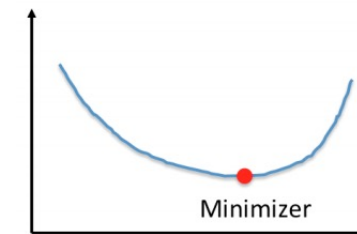
$$f(w_T) - f(w_*) \leq \epsilon$$

w_* : optimal solution
 $f(w_*) = \min_w f(w)$

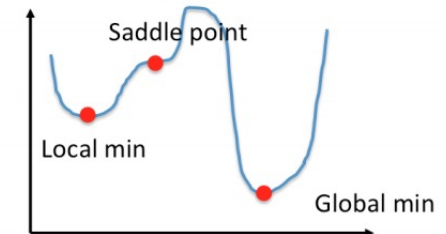
No need to know this solution
Only for convergence analysis



Convex

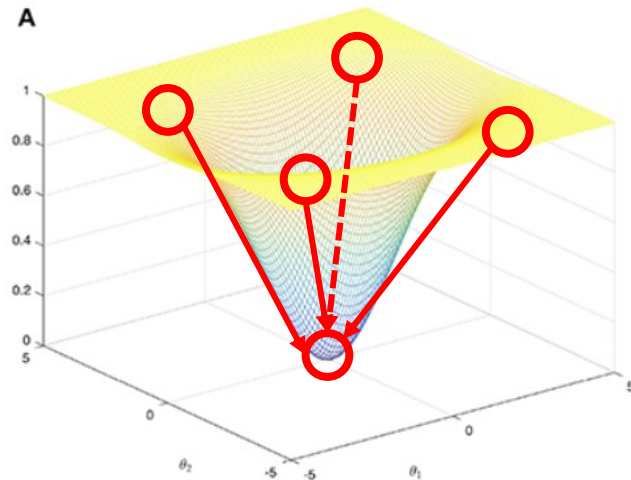


Non-Convex



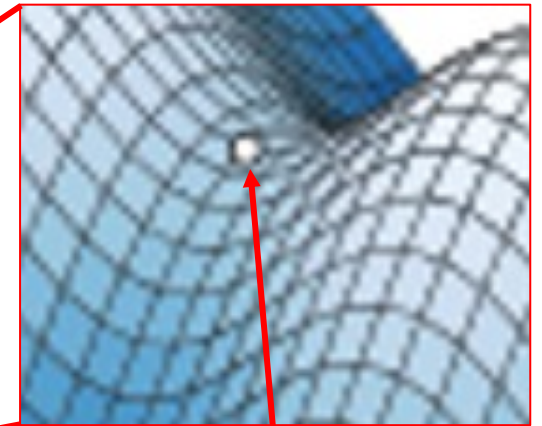
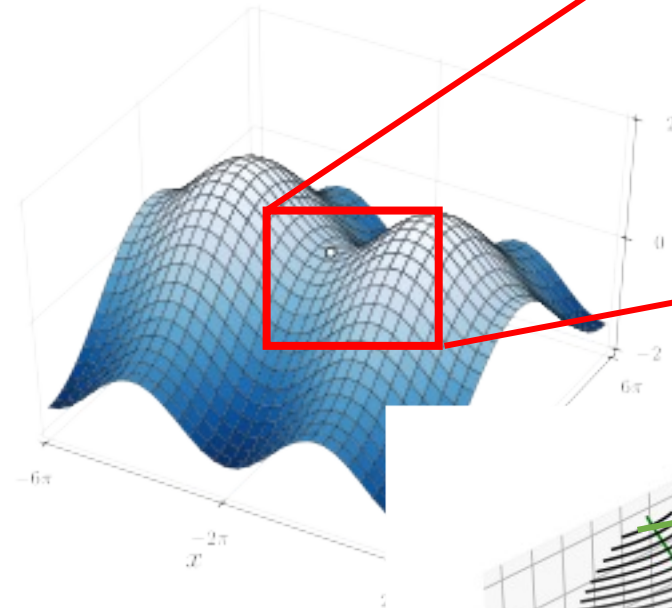
Difficulty of training GANs

- Difficult to converge

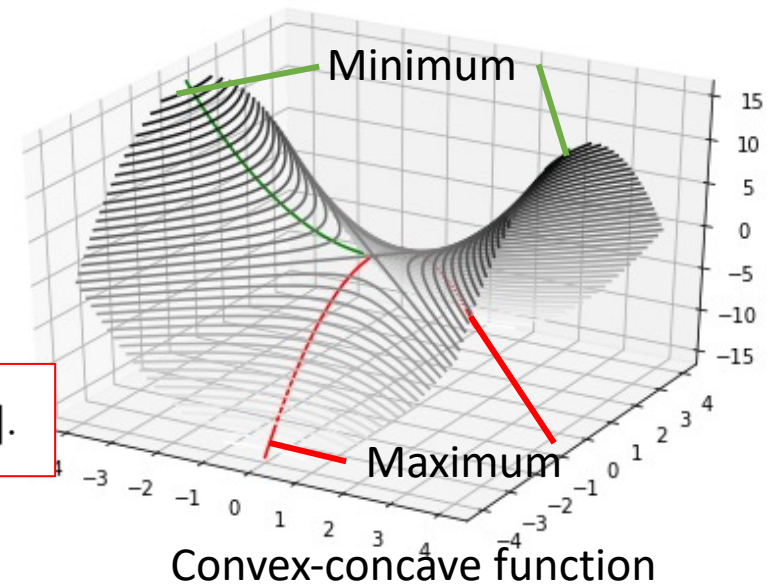


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

nonconvex-nonconcave function
(Hard to define what convergence could be)



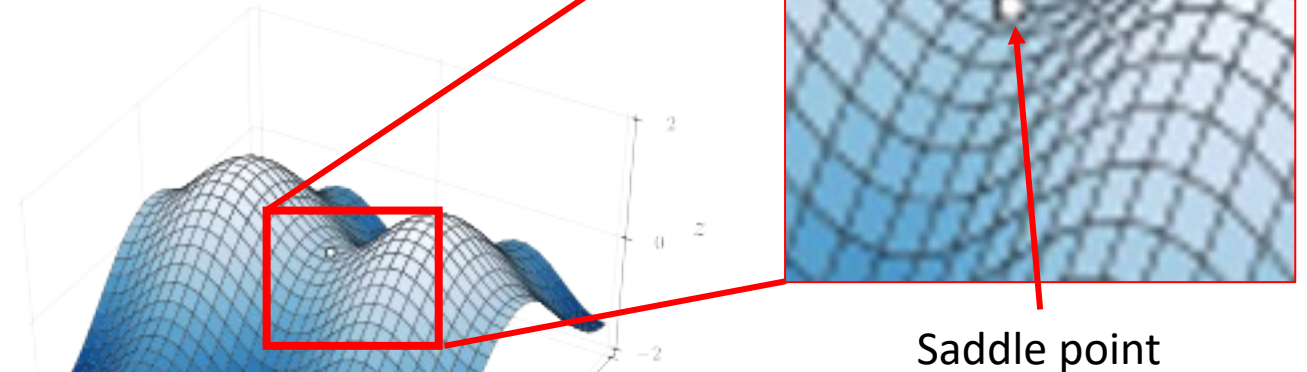
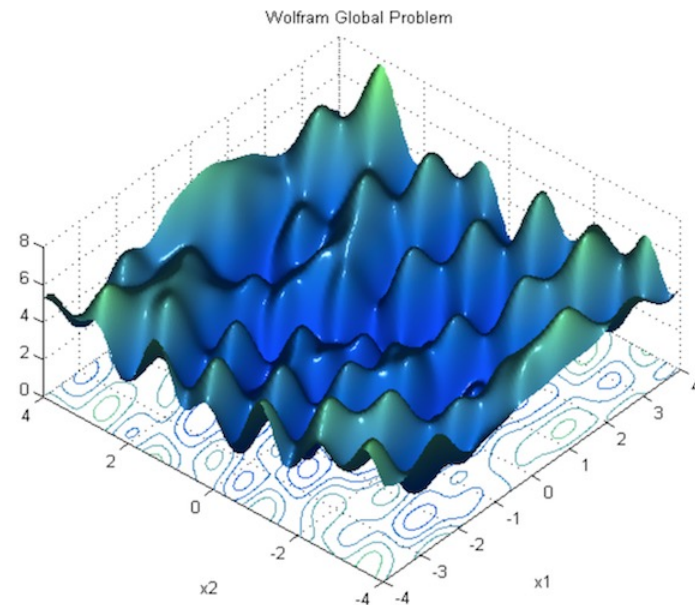
Saddle point



Convex-concave function

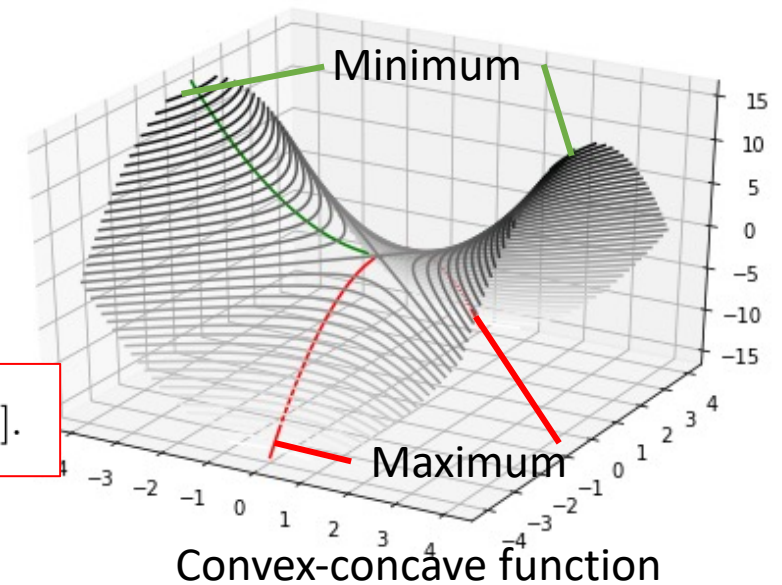
Difficulty of training GANs

- Difficult to converge



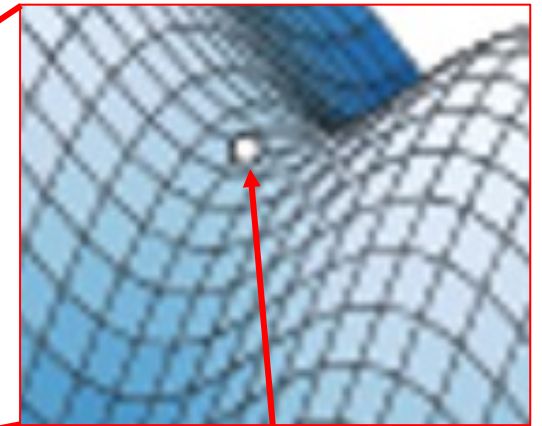
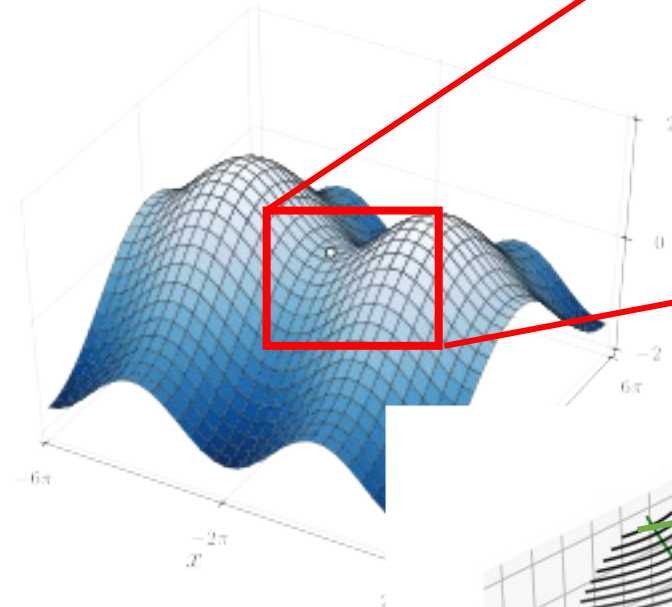
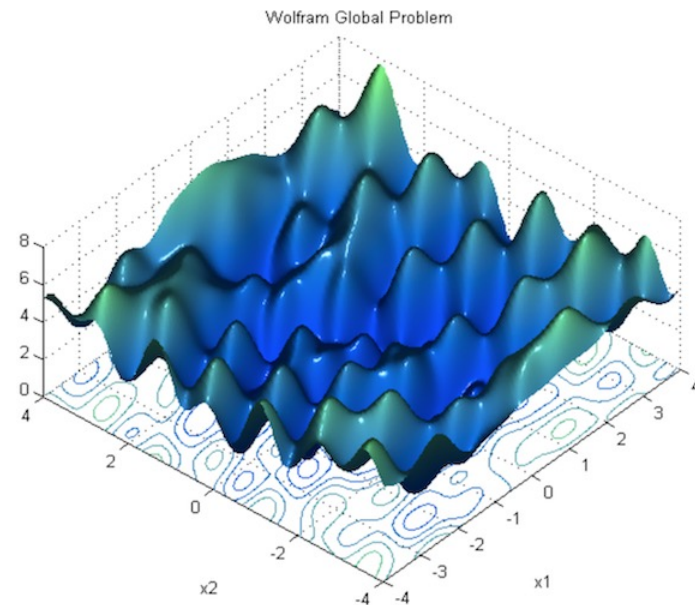
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

nonconvex-nonconcave function
(Hard to define what convergence could be)

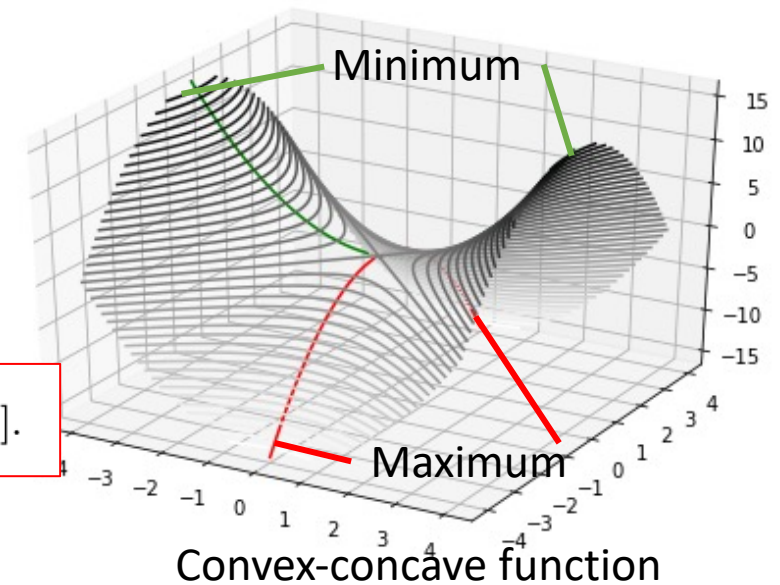


Difficulty of training GANs

- Difficult to converge



Saddle point



Minimum

Maximum

Convex-concave function

We even do not
have a good
accuracy measure
for convergence

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

nonconvex-nonconcave function

(Hard to define what convergence could be)

GAN convergence

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

GAN convergence

Q: what do we expect for the convergence of this objective?

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

GAN convergence

Q: what do we expect for the convergence of this objective?

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

GAN convergence

Q: what do we expect for the convergence of this objective?

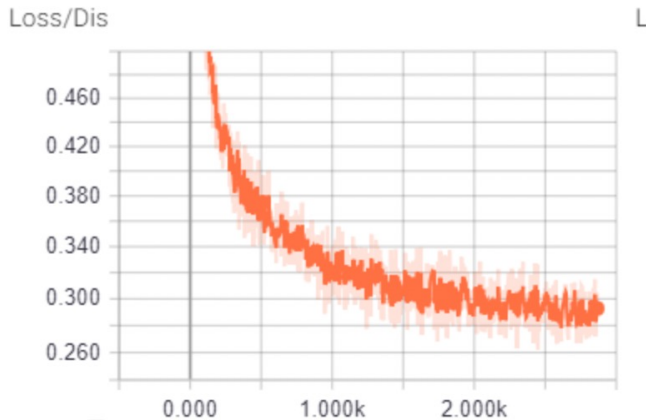
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))].$$

GAN convergence

Q: what do we expect for the convergence of this objective?

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

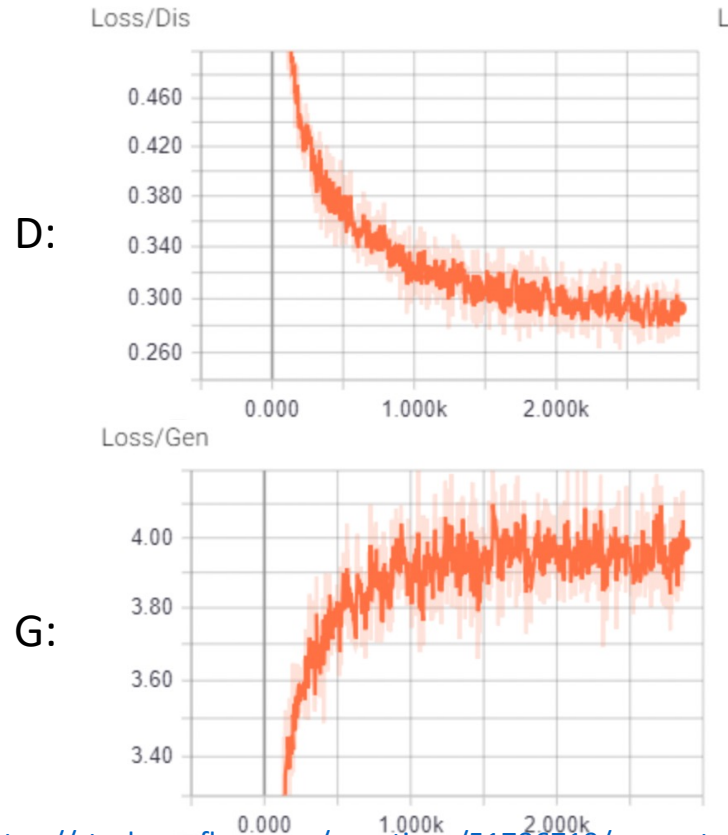
D:



GAN convergence

Q: what do we expect for the convergence of this objective?

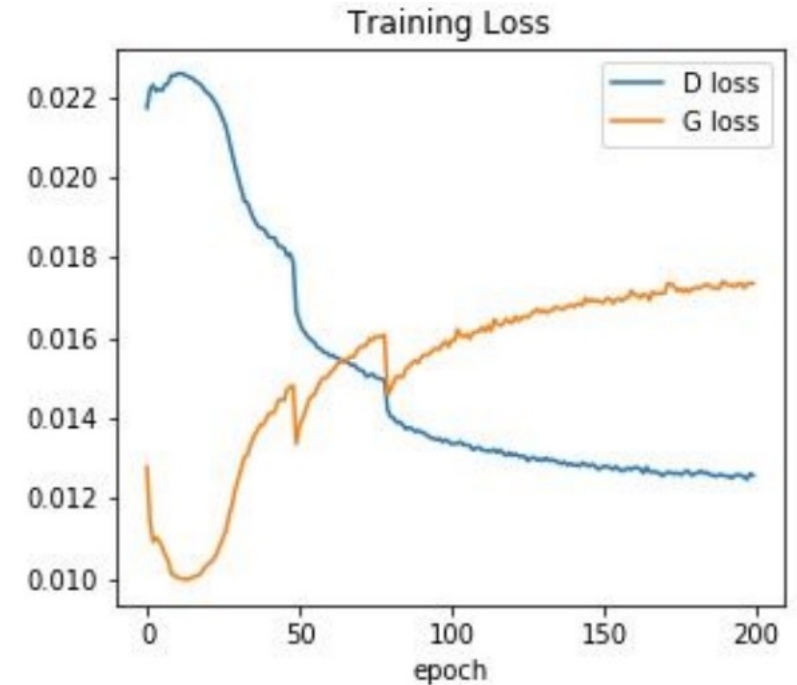
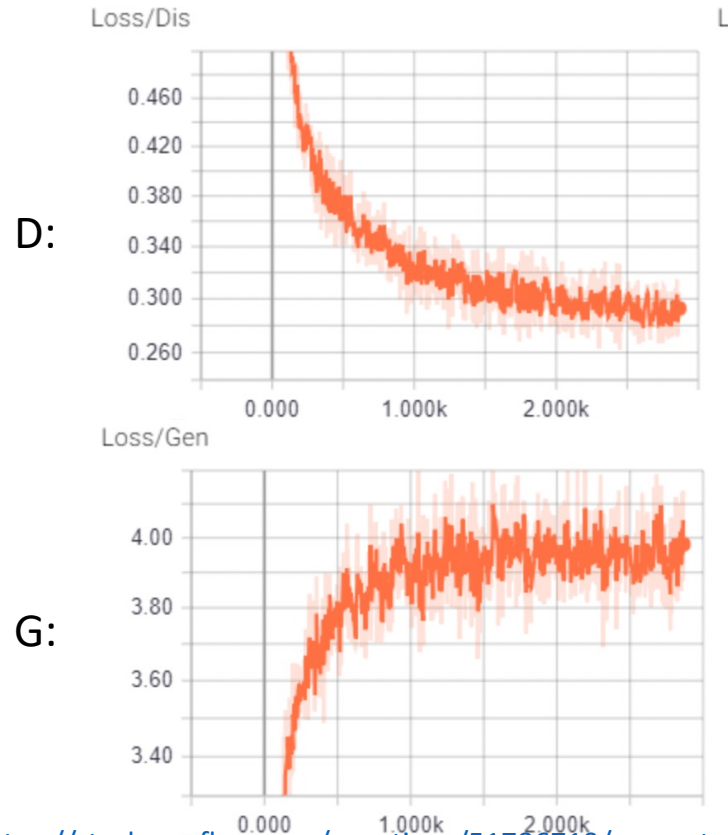
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$



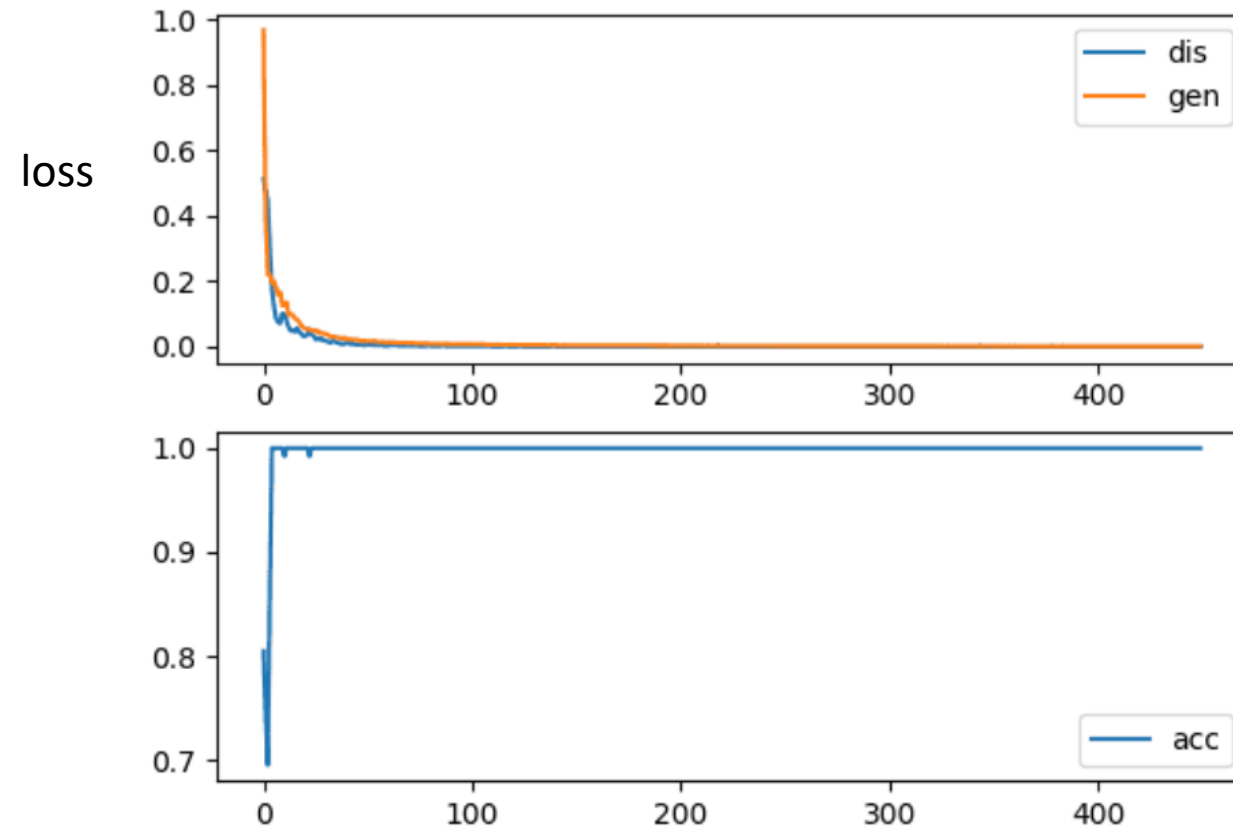
GAN convergence

Q: what do we expect for the convergence of this objective?

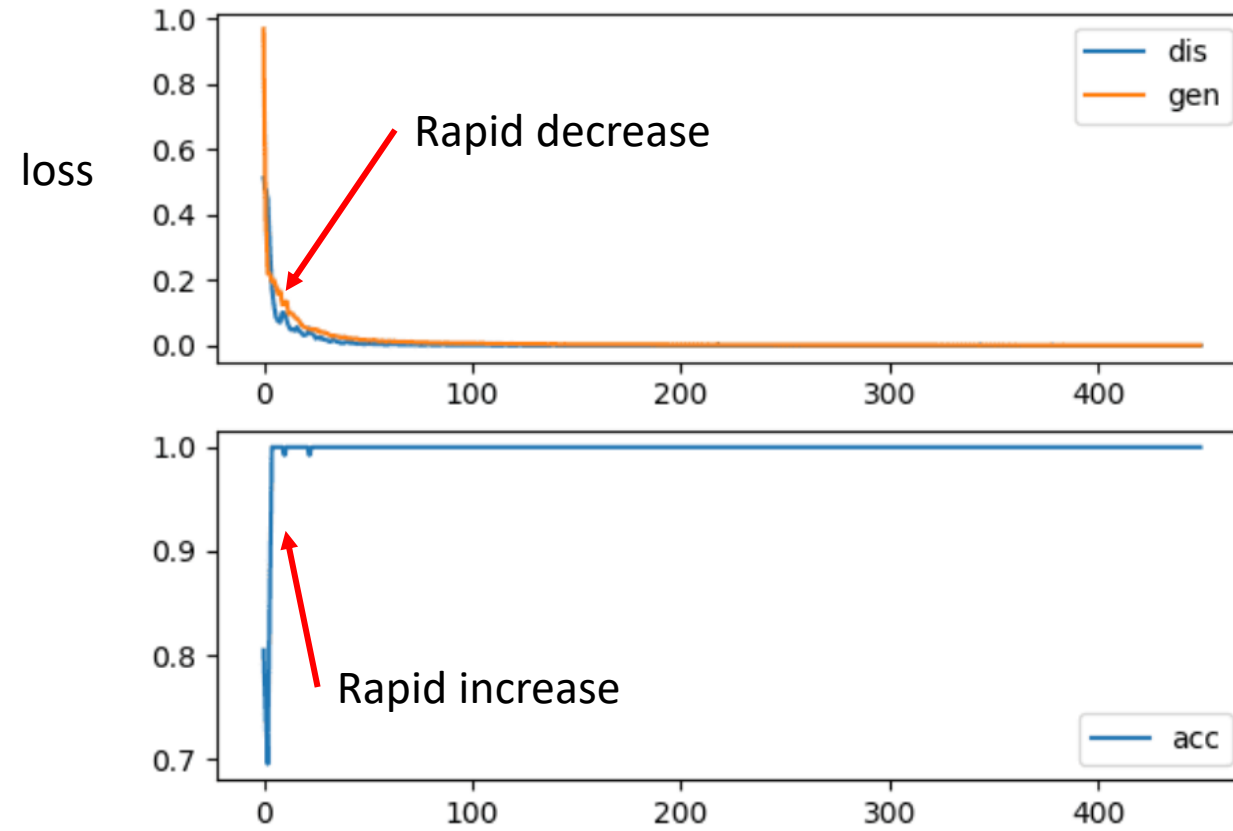
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$



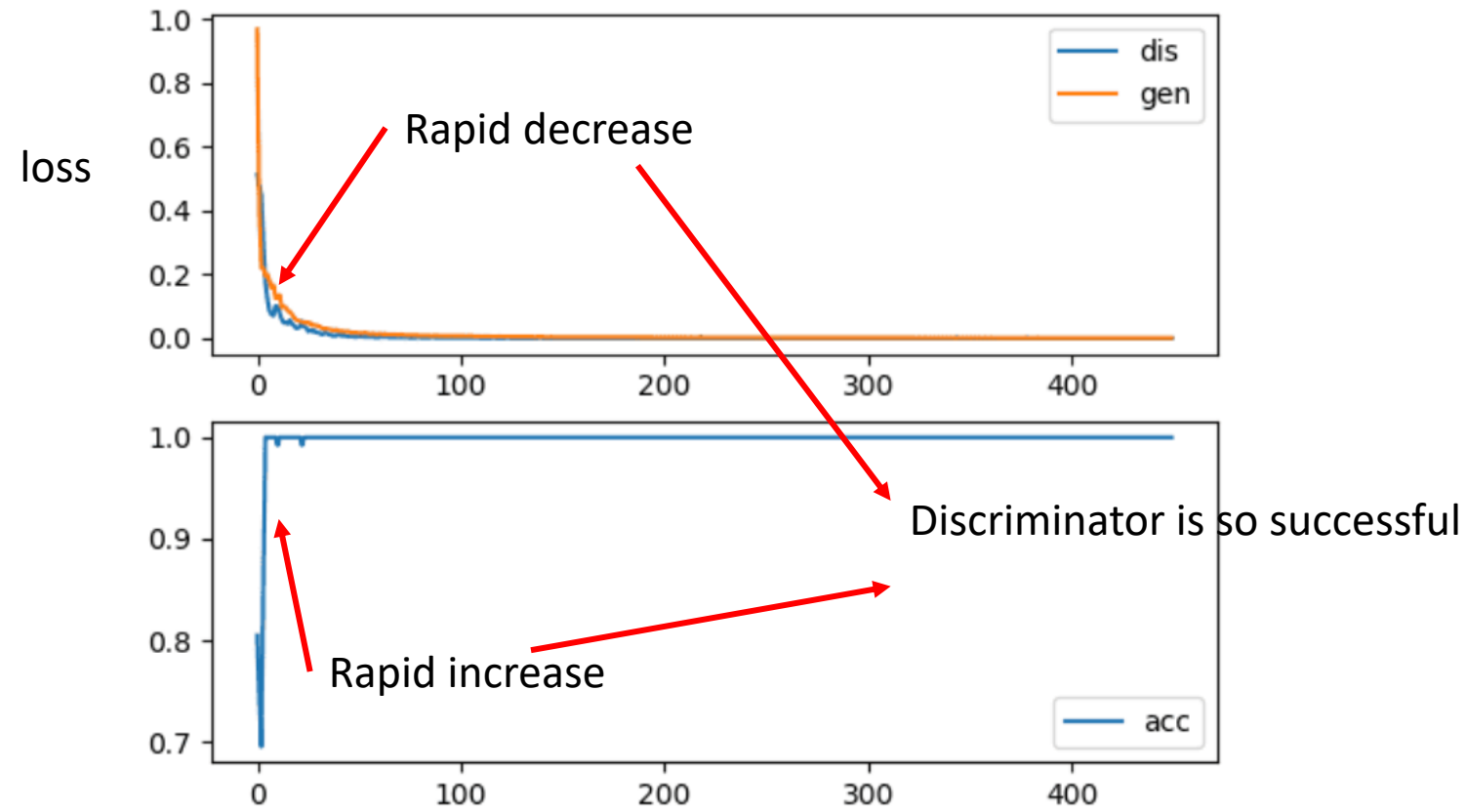
Convergence failure



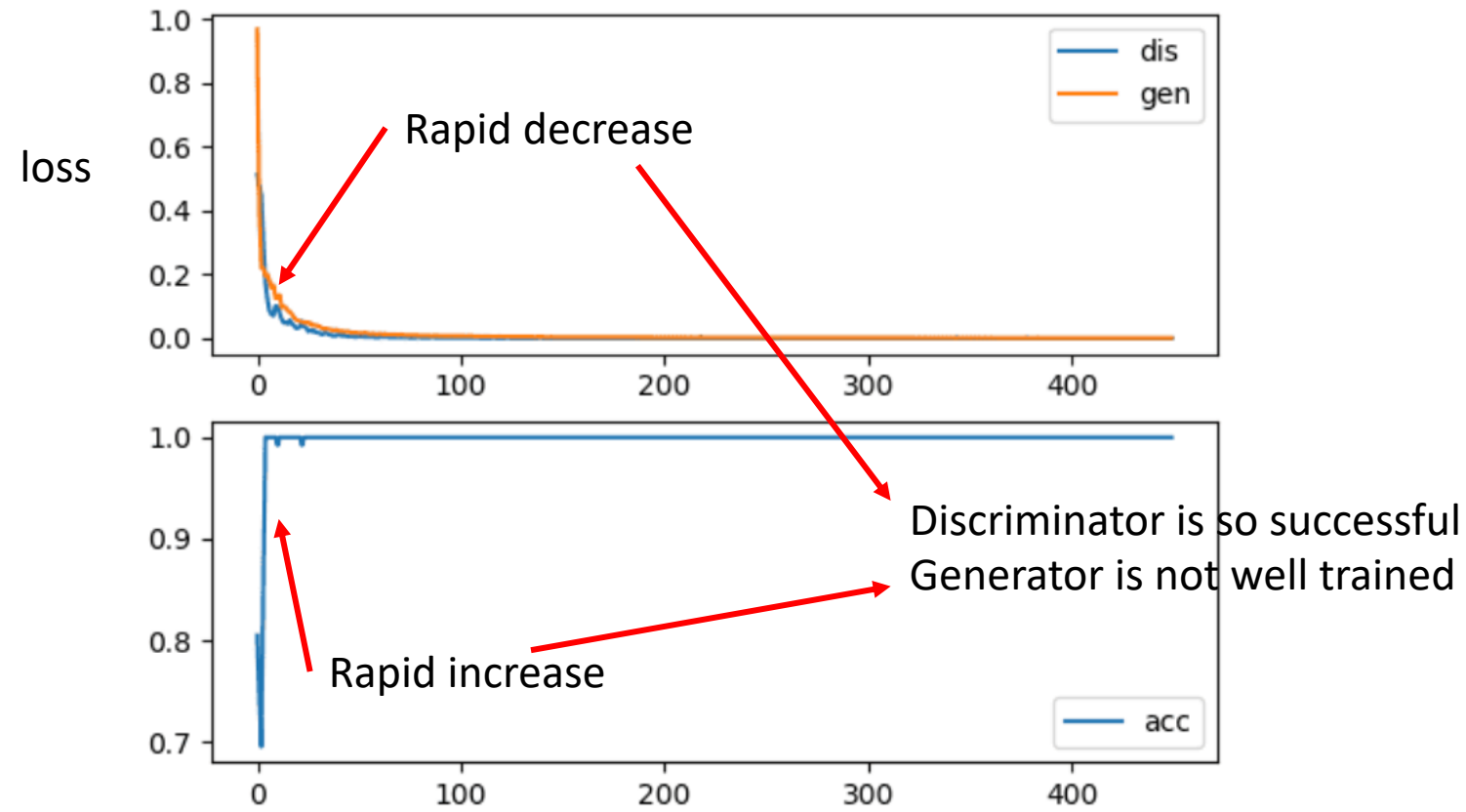
Convergence failure



Convergence failure

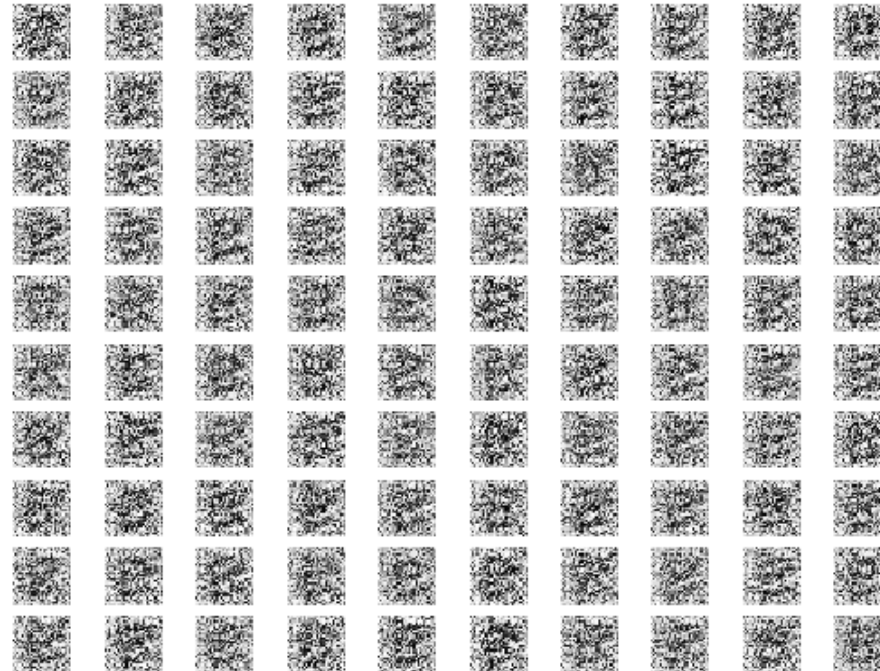


Convergence failure



Convergence failure

loss

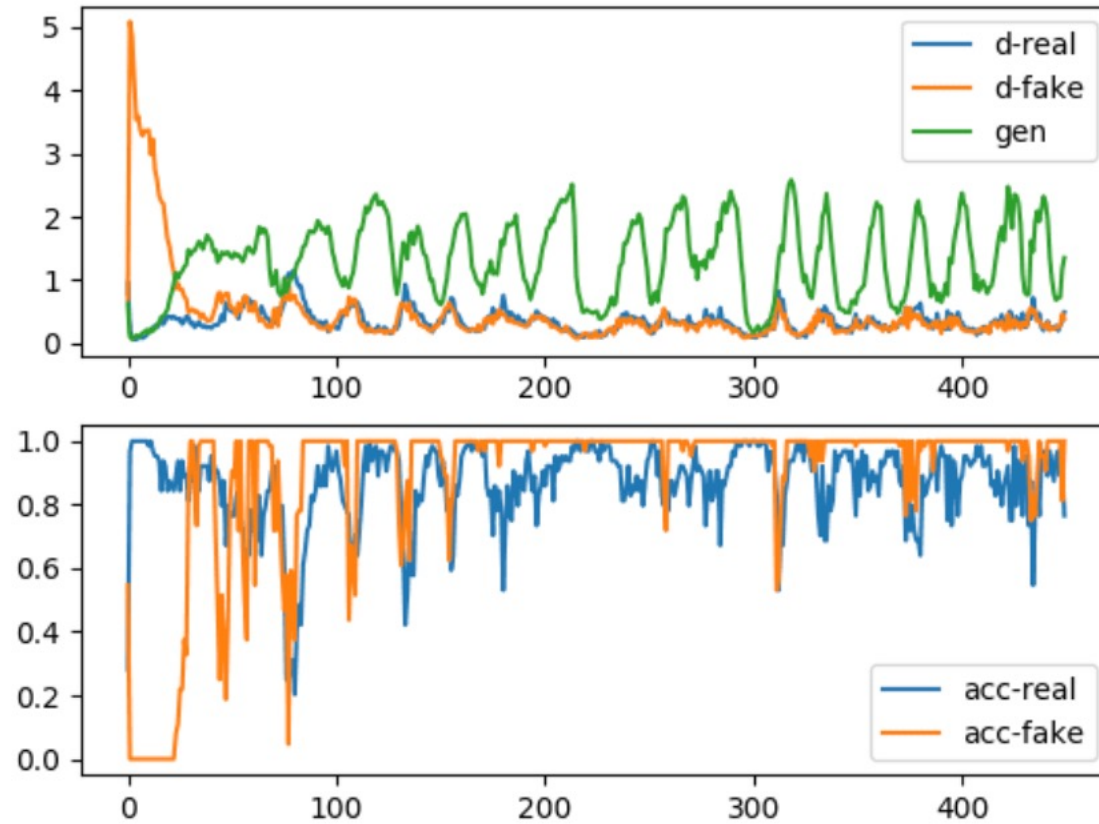


Sample of 100 Generated Images of a **Handwritten Number 8** at Epoch 450 From a GAN That Has a Convergence Failure via Combined Updates to the Discriminator.

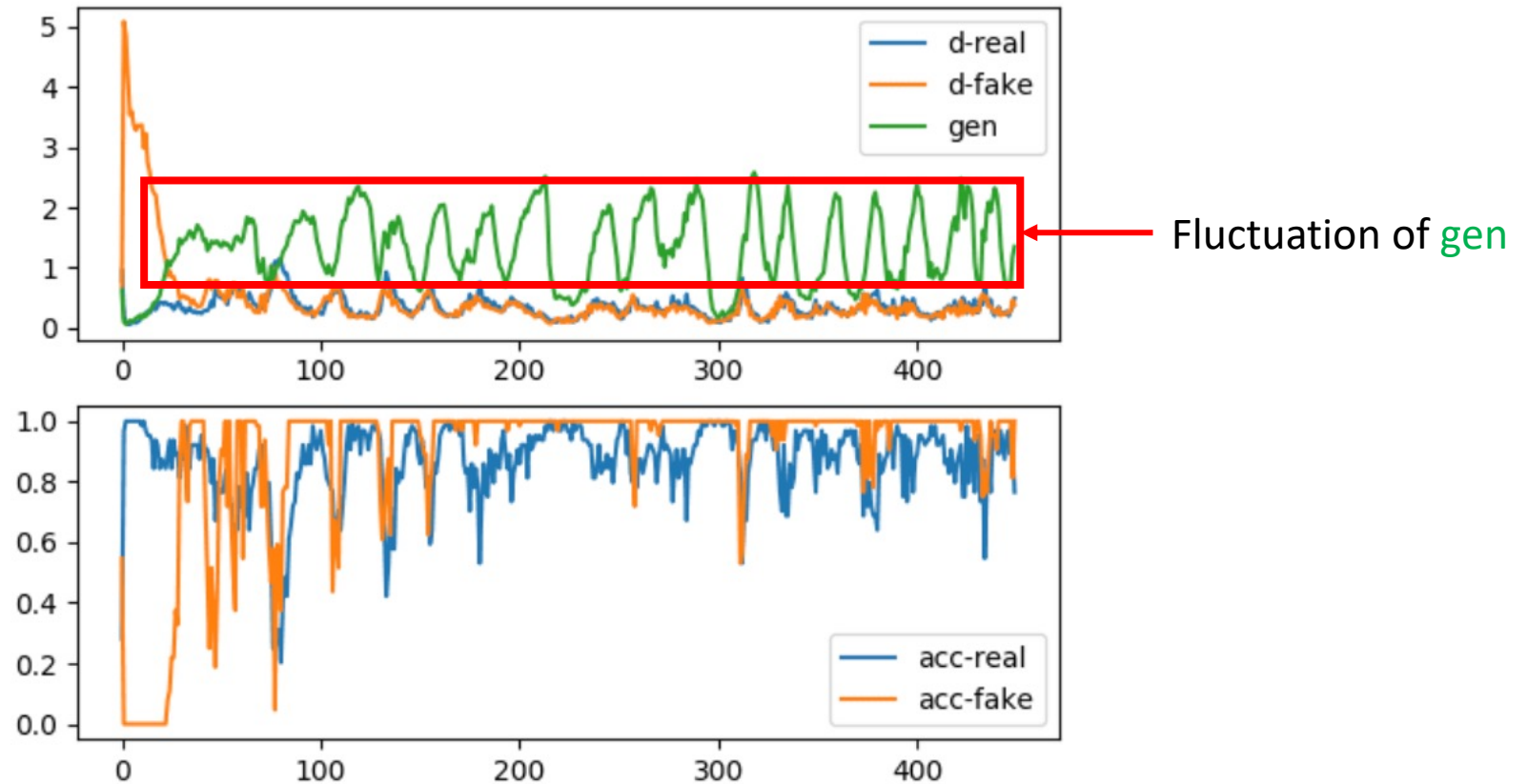
Difficulty of training GANs

- Difficult to converge
- Mode collapse: the generator collapses which produces limited varieties of samples

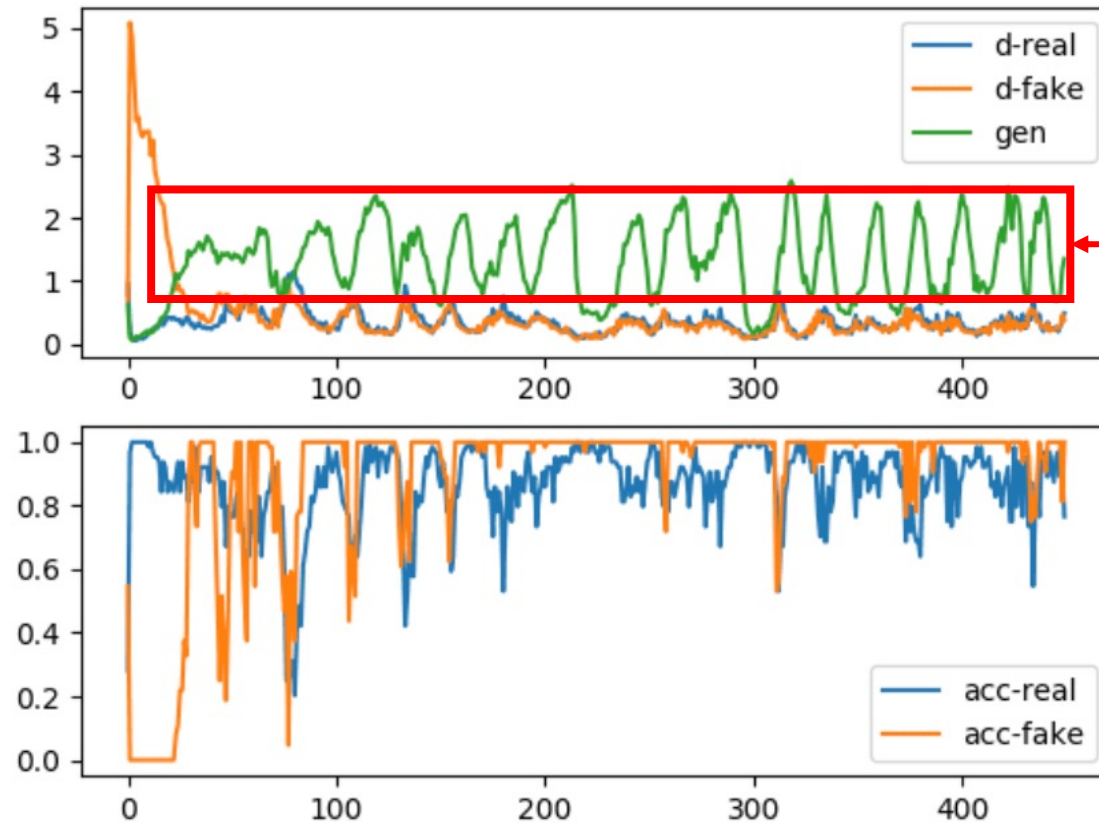
Mode collapse



Mode collapse



Mode collapse



Fluctuation of **gen**



z is different, but generates identical output

Difficulty of training GANs

- Difficult to converge
- Mode collapse: the generator collapses which produces limited varieties of samples
- Unbalance between the generator and discriminator causing overfitting
-

Application of GANs as generative models



Application of GANs as generative models

Q: which images are generated?



Application of GANs as generative models

Q: what about these?



Application of GANs as generative models

Q: what about these?

All are generated by GAN



Application of GANs as generative models

Q: what about these?

All are generated by GAN



Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401-4410. 2019.

<https://arxiv.org/pdf/1812.04948.pdf>



Application of GANs as generative models



Application of GANs as generative models



Application of GANs as generative models



Application of GANs as generative models



Application of GANs as generative models



Monet → photo



photo → Monet

Application of GANs as generative models



Monet → photo



photo → Monet



Photograph



Monet



Van Gogh



Cezanne



Ukiyo-e

Application of GANs as generative models



Monet → photo



photo → Monet



Photograph



Monet



Van Gogh



Cezanne



Ukiyo-e

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In *Proceedings of the IEEE international conference on computer vision*, pp. 2223-2232. 2017.

<https://arxiv.org/pdf/1703.10593.pdf>