# CS 446/ECE 449: Machine Learning

Lecture 21: Generative Adversarial Networks

Han Zhao
04/04/2024

# Today

- TV distance & Jensen-Shannon divergence
- Generative Adversarial Networks
- Gradient Descent-Ascent algorithm

# Recap: Kullback-Leibler divergence

How to measure the discrepancy between
two distributions, $P, Q$?

Kullback-Leibler divergence (KL-div, aka relative entropy):

Discrete: $$D_{\mathrm{KL}}(P \parallel Q) := \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right)$$

Continuous: $$D_{\mathrm{KL}}(P \parallel Q) := \int_{-\infty}^{\infty} p(x) \log \left( \frac{p(x)}{q(x)} \right) dx$$

Notes:

- Always non-negative
- Not symmetric, hence not a distance
- No upper bound in general (think about one example)

# Recap: Total Variation

Total variation (TV) distance:

$$d_{\mathrm{TV}}(P, Q) := \sup_{A \subseteq \mathbb{R}^d} |P(A) - Q(A)|$$

**Proposition:**

Let $P, Q$ be two probability distributions over the same space, then

$$d_{\mathrm{TV}}(P, Q) = \frac{1}{2} \sum_{x \in \mathcal{X}} |P(x) - Q(x)| =: \frac{1}{2} \|P - Q\|_1$$

Proof:

Let $A^*$ be the event such that $d_{\mathrm{TV}}(P, Q) = P(A^*) - Q(A^*)$. Then

$$\forall x \in \mathcal{X} \backslash A^*, \; P(x) < Q(x)$$

Hence,

$$\begin{aligned}
\|P - Q\|_1 &= \sum_{x \in \mathcal{X}} |P(x) - Q(x)| \\
&= \sum_{x \in A^*} (P(x) - Q(x)) + \sum_{x \in \mathcal{X} \backslash A^*} (Q(x) - P(x)) \\
&= (P(A^*) - Q(A*)) + (1 - Q(A^*) - (1 - P(A^*))) \\
&= 2\,(P(A^*) - Q(A*)) = 2 d_{\mathrm{TV}}(P, Q)
\end{aligned}$$

# Recap: Total Variation

Total variation (TV) distance:

$$d_{\mathrm{TV}}(P, Q) := \sup_{A \subseteq \mathbb{R}^d} |P(A) - Q(A)|$$

Interpretation of the TV-distance as a binary classification problem:

- Let $P, Q$ be two known distributions
- An adversary chooses a distribution $D$ as follows:

$$D = \begin{cases} P & w.p.\ 0.5 \\ Q & w.p.\ 0.5 \end{cases}$$

- You get to see a sample from $D$, and then make a (possibly randomized) guess about whether $D$ is $P$ or $Q$?
- Given knowledge about $P, Q$, upon receiving the sample $x$, what's your best strategy under the 0-1 binary error?

# Recap: Total Variation

Interpretation of the TV-distance as a binary classification problem:

$$\eta(x) := \Pr(\text{my guess is } P \mid x)$$

Then, the error probability of using strategy $\eta(\cdot)$ when seeing $x$ is:

$$\begin{aligned}\Pr(\text{error}, x) &= \Pr(\text{guess } P, x \sim Q) + \Pr(\text{guess } Q, x \sim P) \\ &= \frac{1}{2}Q(x)\eta(x) + \frac{1}{2}P(x)(1-\eta(x))\end{aligned}$$

Hence, the overall error probability is given by:

$$\begin{aligned}\Pr(\text{ error }) &= \sum_{x\in\mathcal{X}} \Pr(\text{ error }, x) \\ &= \sum_{x\in\mathcal{X}} \frac{1}{2}Q(x)\eta(x) + \frac{1}{2}P(x)(1-\eta(x)) \\ &= \frac{1}{2} + \frac{1}{2}\sum_{x\in\mathcal{X}} \eta(x)(Q(x)-P(x))\end{aligned}$$

# Recap: Total Variation

Interpretation of the TV-distance as a binary classification problem:

$$\eta(x) := \Pr(\text{my guess is } P \mid x)$$

Hence, the overall error probability is given by:

$$\Pr(\text{error}) = \frac{1}{2} + \frac{1}{2} \sum_{x \in \mathcal{X}} \eta(x) \left(Q(x) - P(x)\right)$$

So, clearly, to minimize the overall guessing error, the best strategy is given by:

$$\eta(x) = \begin{cases} 1 & P(x) \geq Q(x) \\ 0 & P(x) < Q(x) \end{cases}$$

Under this optimal strategy, the optimal distinguishing error is:

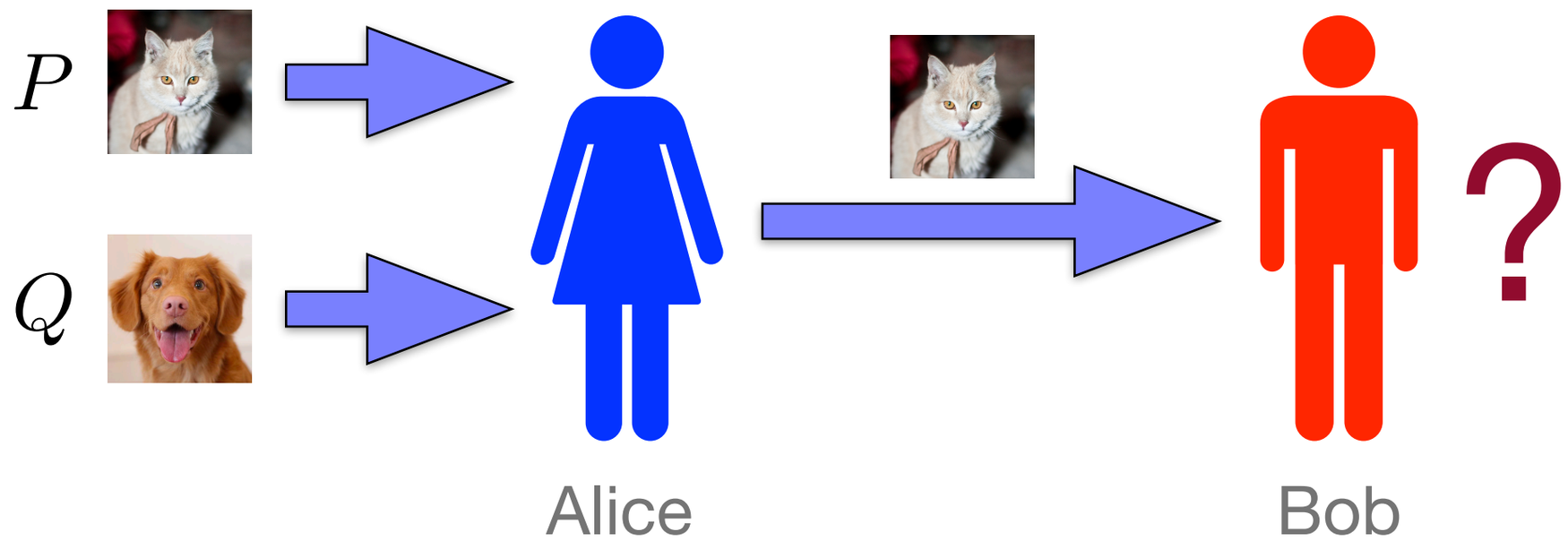
$$\begin{aligned} \Pr(\text{error}) &= \frac{1}{2} + \frac{1}{2} \sum_{x : P(x) \geq Q(x)} Q(x) - P(x) \\ &= \frac{1}{2} - \frac{1}{2} d_{\mathrm{TV}}(P, Q) \end{aligned}$$

Consider some extremal cases of this distinguishing game

# Recap: Total Variation

## Total Variation (TV) distance & the distinguishing game

TV distance: $d_{\mathrm{TV}}(P, Q) := \sup_{A \subseteq \mathbb{R}^d} |P(A) - Q(A)|$




**Bob's optimal error:** $\Pr(\text{error}) = \frac{1}{2} + \frac{1}{2} \sum_{x: P(x) \geq Q(x)} Q(x) - P(x)$

$$= \frac{1}{2} - \frac{1}{2} d_{\mathrm{TV}}(P, Q)$$

# Jensen-Shannon divergence

Let's revisit our binary distinguishing game:

- Let $P, Q$ be two known distributions
- An adversary chooses a distribution $D$ as follows:

$$D = \begin{cases} P & w.p.\ 0.5 \\ Q & w.p.\ 0.5 \end{cases}$$

- You get to see a sample from $D$, and then make a (possibly randomized) guess about whether $D$ is $P$ or $Q$?
- Given knowledge about $P, Q$, upon receiving the sample $x$, what's your best strategy under the cross-entropy loss?

# Jensen-Shannon divergence

Recall the cross entropy loss:

$$\ell_{\mathrm{CE}}(\hat{y}, y) = -y \log \hat{y} - (1-y)\log(1-\hat{y})$$

- True label $y \in \{0,1\}$
- Predicted probability $\hat{y} \in [0,1]$
- Note that the loss is still well-defined for soft label $y \in [0,1]$
- In the case of hard label $y \in \{0,1\}$, it is the same as the log-loss used in logistic regression, a.k.a. negative conditional log-likelihood
- It can also be understood as the KL-divergence (up to a constant) between the true Bernoulli distribution $\mathrm{Ber}(y)$ and the predicted Bernoulli distribution $\mathrm{Ber}(\hat{y})$

$$\ell_{\mathrm{CE}}(\hat{y}, y) = H(\mathrm{Ber}(y)) + D_{\mathrm{KL}}(\mathrm{Ber}(y) \| \mathrm{Ber}(\hat{y}))$$

# Jensen-Shannon divergence

Let's re-do the analysis under the cross-entropy loss instead:

$$\eta(x) := \Pr(\text{my guess is } P \mid x)$$

Then, the cross-entropy loss of using strategy $\eta(\cdot)$ when seeing $x$ is:

$$\ell^x_{\text{CE}} = -\frac{1}{2}P(x)\log\eta(x) - \frac{1}{2}Q(x)\log(1-\eta(x))$$

(HW 5): Hence, under the given $x$, the optimal choice of $\eta(x)$ will be:

$$\eta(x) = \frac{P(x)}{P(x)+Q(x)}$$

Note:

- Different from the 0-1 loss, in this case the optimal strategy is randomized, not deterministic
- Only when $P(x) = 0$ or $Q(x) = 0$, the optimal strategy becomes deterministic

# Jensen-Shannon divergence

Let's re-do the analysis under the cross-entropy loss instead:

$$\eta(x) := \Pr(\text{my guess is } P \mid x)$$

Then, the cross-entropy loss of using strategy $\eta(\cdot)$ when seeing $x$ is:

$$\ell^x_{\text{CE}} = -\frac{1}{2}P(x)\log\eta(x) - \frac{1}{2}Q(x)\log(1-\eta(x))$$

(HW 5): Hence, under the given $x$, the optimal choice of $\eta(x)$ will be:

$$\eta(x) = \frac{P(x)}{P(x)+Q(x)}$$

(HW 5): Plug in the optimal (randomized) strategy $\eta(x)$ into the above per-instance loss and sum over all the potential instances, we can show that the final loss will have the following form:

$$\log(2) - D_{\text{JS}}(P;Q)$$

where $D_{\text{JS}}(P;Q) := \frac{1}{2}D_{\text{KL}}\left(P\|\frac{P+Q}{2}\right) + \frac{1}{2}D_{\text{KL}}\left(Q\|\frac{P+Q}{2}\right)$ is called the Jensen-Shannon divergence between $P$ and $Q$.

# Jensen-Shannon divergence

Jensen-Shannon divergence between $P$ and $Q$:

$$D_{\mathrm{JS}}(P;Q) := \frac{1}{2} D_{\mathrm{KL}}\left(P \| \frac{P+Q}{2}\right) + \frac{1}{2} D_{\mathrm{KL}}\left(Q \| \frac{P+Q}{2}\right)$$

Notes:

- Always non-negative due to the non-negativity of the KL divergence
- Symmetric, but still not a metric (does not satisfy the triangle inequality)
- Bounded between $[0, \log 2]$ (under natural logarithm)
- Equals 0 iff $P = Q$ a.s.

Recall the optimal error of our binary distinguishing game under the cross-entropy loss:

$$\log(2) - D_{\mathrm{JS}}(P;Q)$$

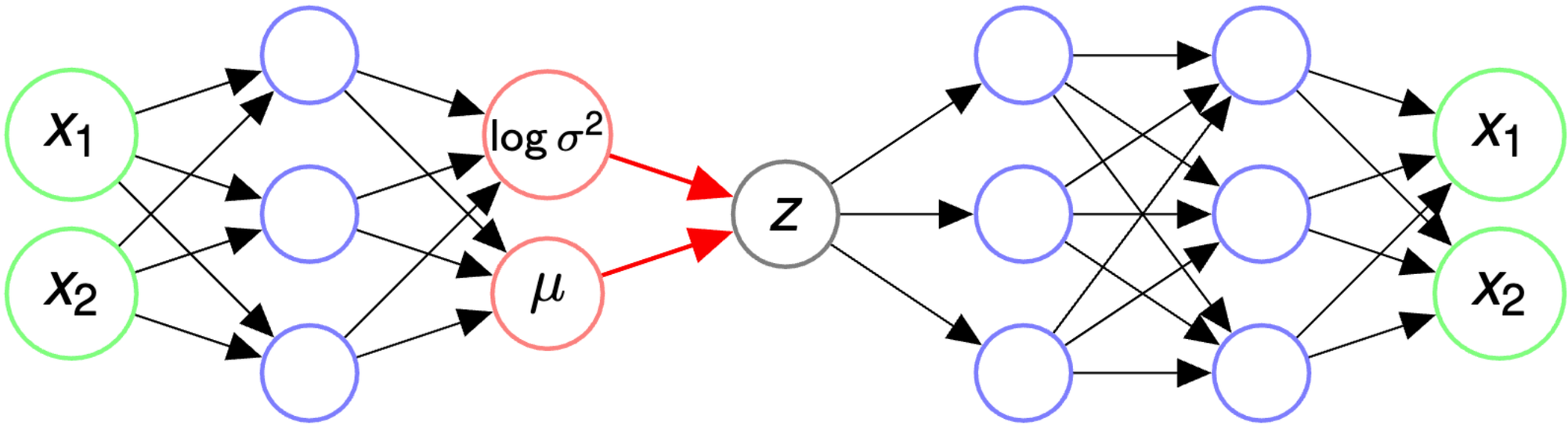
- Always non-negative since $D_{\mathrm{JS}}(P;Q) \leq \log(2)$
- Takes the maximum value of $\log(2)$ iff $P = Q$
- Is 0 if the supports of $P$ and $Q$ are disjoint

# Generative Adversarial Networks

What do we spend so much time discussing the binary distinguishing game?

## Because it lends us a way to build generative models

Recall what we learned from last lecture on using the variational auto-encoder to generate data through reconstruction:




$$\mathcal{L}(p_\theta, q_\phi) \approx -D_{KL}(q_\phi, p) + \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x|z^i)$$

Limitation: we will need to make (independence & parametric) assumptions on $p_\theta(x \mid z)$ in order to make it tractable.

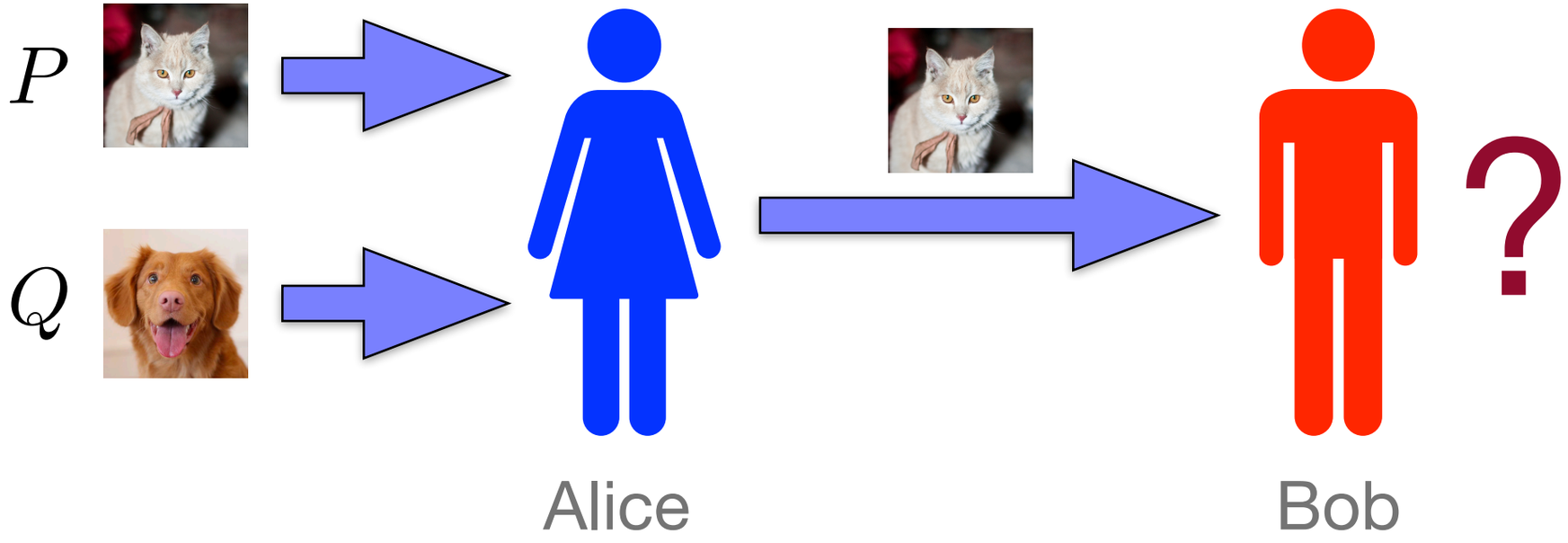
# Generative Adversarial Networks

Let's try to build a generative model such that:

- It is free of independence & parametric assumption on the generated data
- The generative distribution is as close as possible to the underlying data distribution

How to measure the distance of two distributions: $p_g$ and $p_d$:

- The generative distribution: $p_g$
- The underlying data distribution: $p_d$




Inspired by the binary distinguishing game, we say $p_g$ and $p_d$ are close if no matter what strategy Bob uses to distinguish the two, he cannot successfully win the game with high accuracy

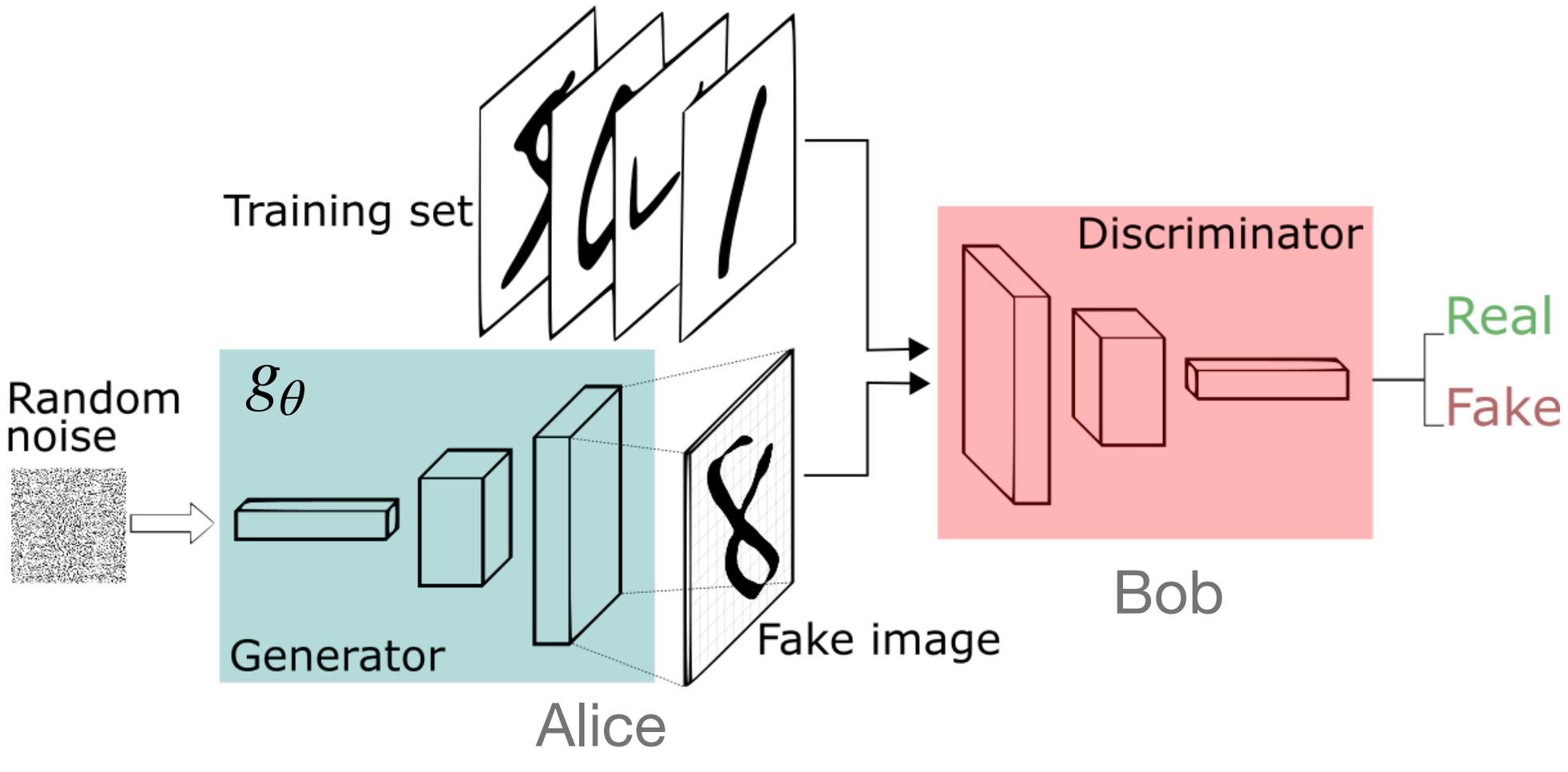
# Generative Adversarial Networks

Let's try to build a generative model such that:

- It is free of independence & parametric assumption on the generated data
- The generative distribution is as close as possible to the underlying data distribution

How to define and construct $p_g$?

- Use a neural network $g_\theta : \mathbb{R}^k \rightarrow \mathbb{R}^d$ with parameter $\theta$ to transform a $d$ -dim noise into a realistic sample



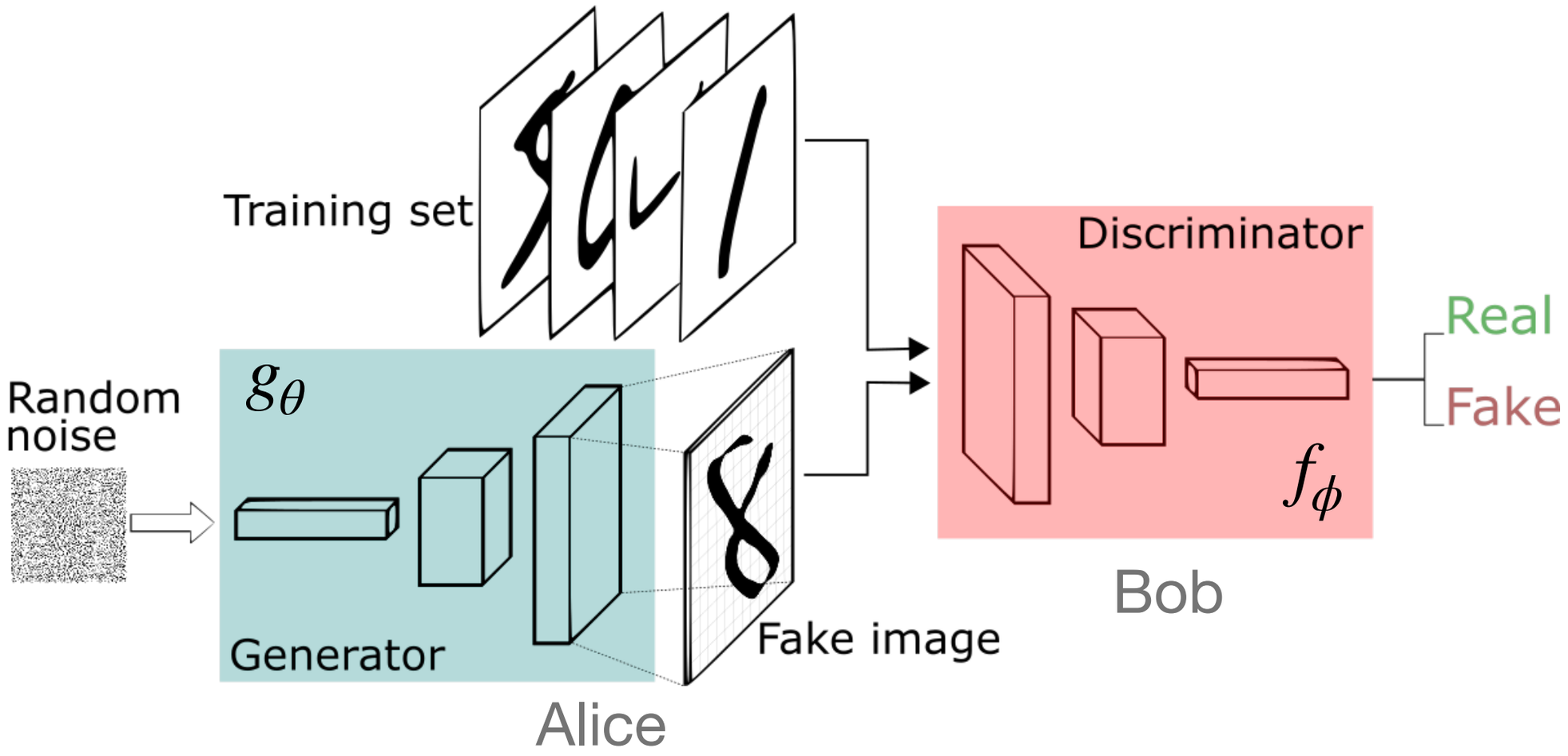
# Generative Adversarial Networks

Let's try to build a generative model such that:

- It is free of independence & parametric assumption on the generated data
- The generative distribution is as close as possible to the underlying data distribution

How to make sure the discriminator (Bob) can try a diverse set of strategies?

- Use a neural network $f_\phi : \mathbb{R}^d \to [0,1]$ with learnable parameter $\phi$ to implement the (randomized) discriminator

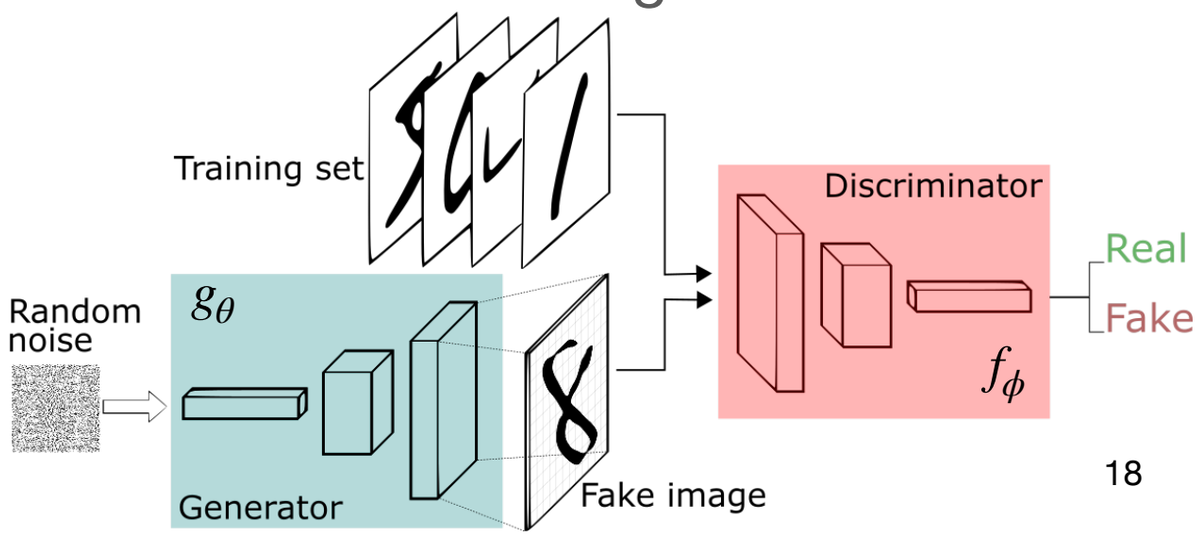


# Generative Adversarial Networks

Combine everything together, we arrive at the following minimax optimization problem (adversarial training)

$$\min_{\theta} \max_{\phi} V(g_\theta, f_\phi) := \mathbb{E}_{x \sim p_d}[\log f_\phi(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - f_\phi(g_\theta(z)))]$$

- We use $z \sim p_z(z)$ to denote the input random noise with under a given (simple) distribution $p_z(\cdot)$, e.g., white Gaussian noise
- The generative distribution $p_g$ is not explicitly defined, but we can sample from it by
  - Sample random noise $z \sim p_z(z)$
  - Transform the noise $z$ into a potentially realistic sample $\hat{x} = g_\theta(z)$
- Again, one can view the minimax objective as a game between the generator and the discriminator, where $V(\cdot, \cdot)$ denotes the value of the game



# Gradient Descent-Ascent Algorithm

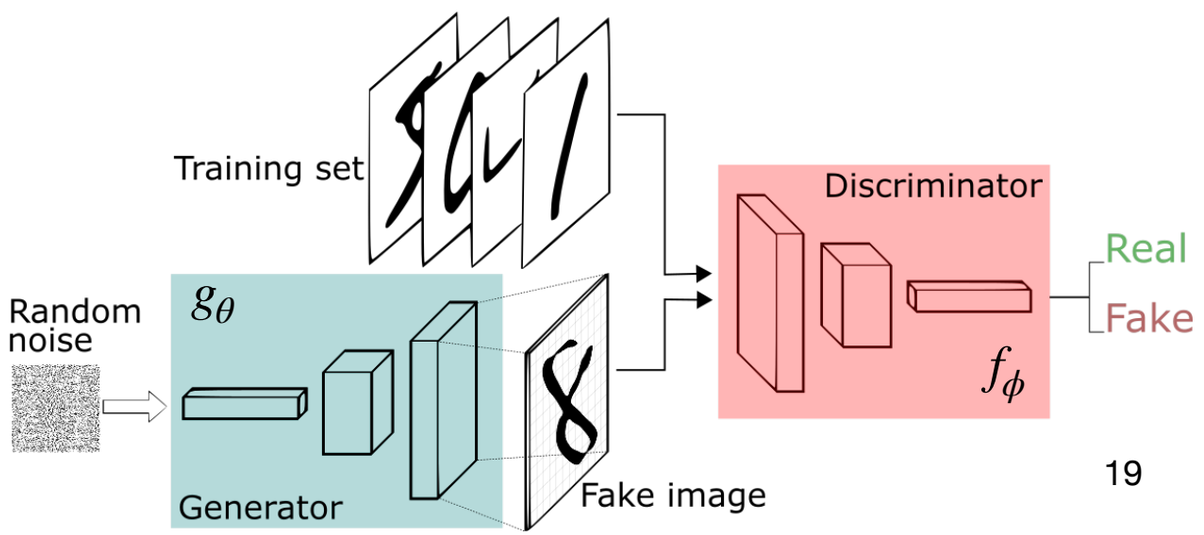
Combine everything together, we arrive at the following minimax optimization problem (adversarial training)

$$\min_{\theta} \max_{\phi} V(g_\theta, f_\phi) := \mathbb{E}_{x \sim p_d}[\log f_\phi(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - f_\phi(g_\theta(z)))]$$

How to optimize the above minimax objective function?

The gradient descent-ascent algorithm (synchronous version)

- Let $t$ be the iteration number and $\gamma > 0$ be the step-size
- Descent:
  - Compute $\nabla_\theta^{(t)} := \nabla_\theta V(g_\theta^{(t)}, f_\phi^{(t)})$
  - Update $\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \nabla_\theta^{(t)}$
- Ascent:
  - Compute $\nabla_\phi^{(t)} := \nabla_\phi V(g_\theta^{(t)}, f_\phi^{(t)})$
  - Update $\phi^{(t+1)} \leftarrow \phi^{(t)} + \gamma \nabla_\phi^{(t)}$



# Gradient Descent-Ascent Algorithm

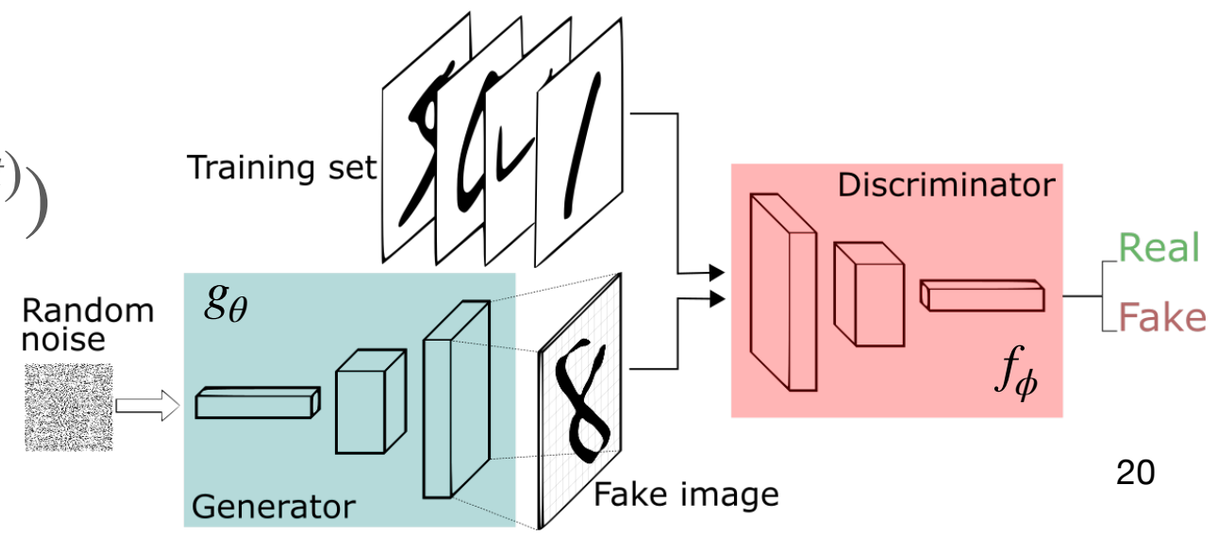
Combine everything together, we arrive at the following minimax optimization problem (adversarial training)

$$\min_{\theta} \max_{\phi} V(g_\theta, f_\phi) := \mathbb{E}_{x \sim p_d}[\log f_\phi(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - f_\phi(g_\theta(z)))]$$

How to optimize the above minimax objective function?

The gradient descent-ascent algorithm (asynchronous version)

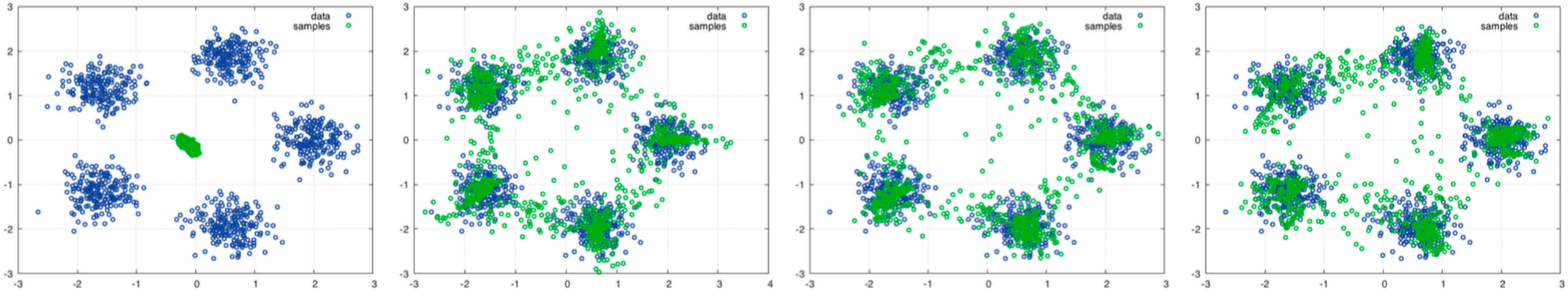
- Let $t$ be the iteration number and $\gamma > 0$ be the step-size
- Descent:
  - Compute $\nabla_\theta^{(t)} := \nabla_\theta V(g_\theta^{(t)}, f_\phi^{(t)})$
  - Update $\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \nabla_\theta^{(t)}$
- Ascent:
  - Compute $\nabla_\phi^{(t)} := \nabla_\phi V(g_\theta^{(t+1)}, f_\phi^{(t)})$
  - Update $\phi^{(t+1)} \leftarrow \phi^{(t)} + \gamma \nabla_\phi^{(t)}$



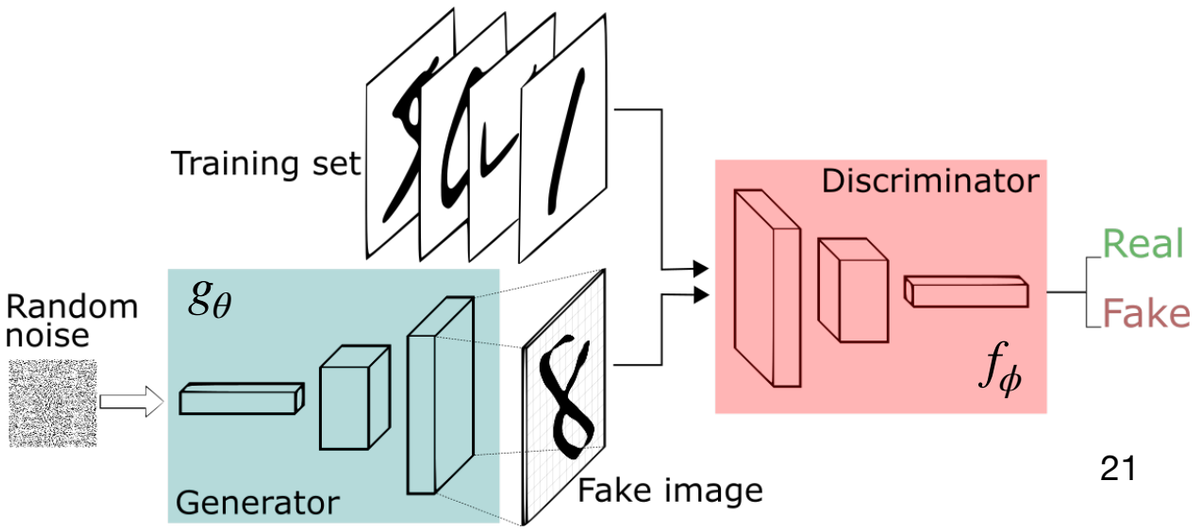
# Generative Adversarial Networks

Some typical examples:

- Synthetic data



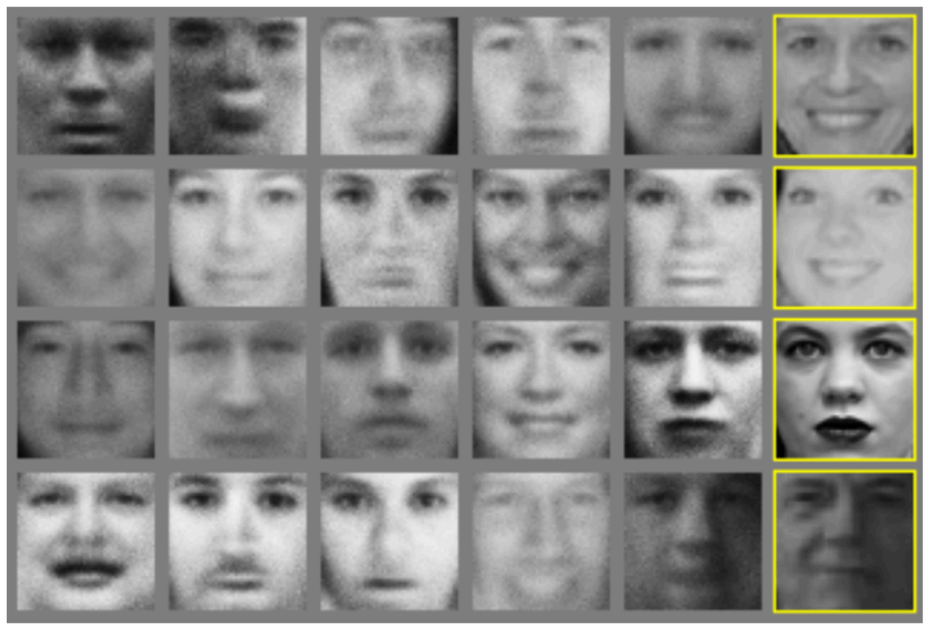

0 Iterations 2000 Iterations 4000 Iterations 5000 Iterations

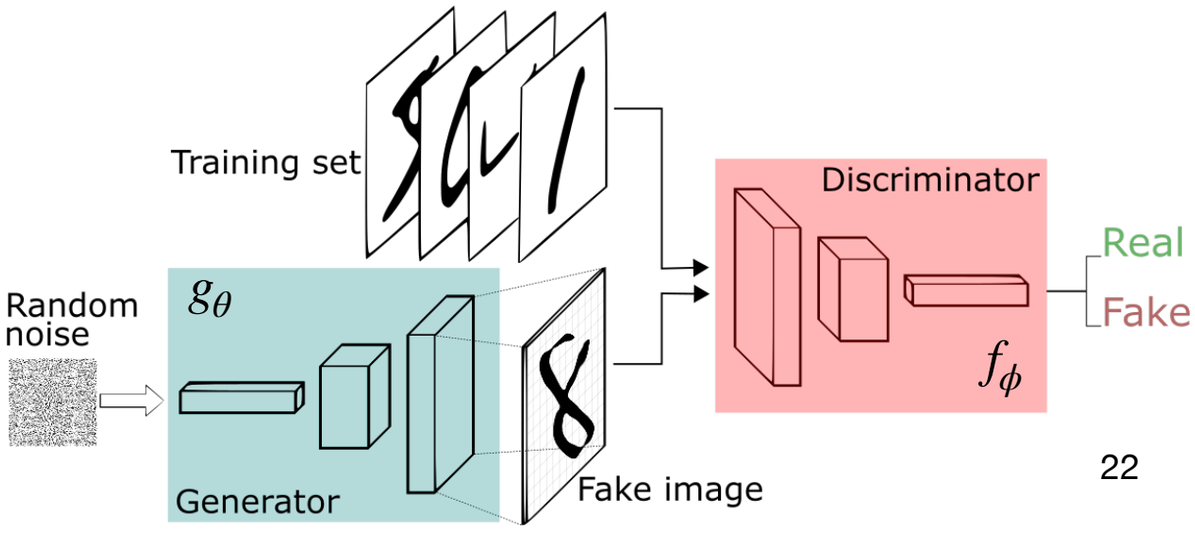


# Generative Adversarial Networks

Some typical examples:

- Image modeling

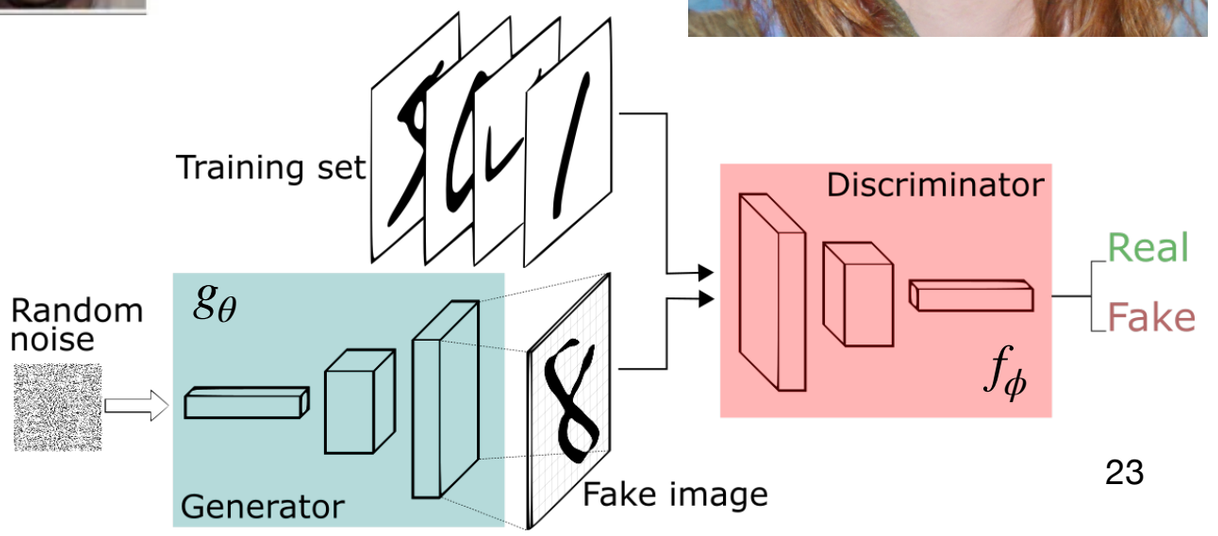




# Generative Adversarial Networks

Some typical examples:

- High-fidelity facial modeling using StyleGAN

# Next Time

- Diffusion Models